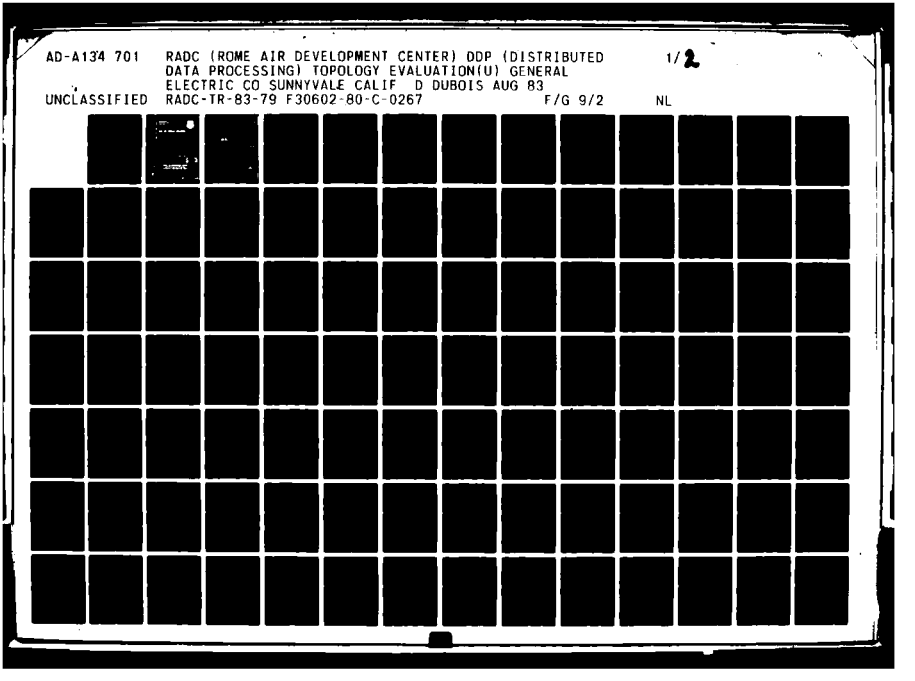
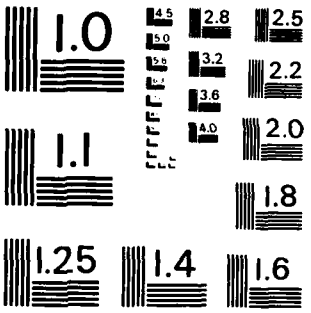
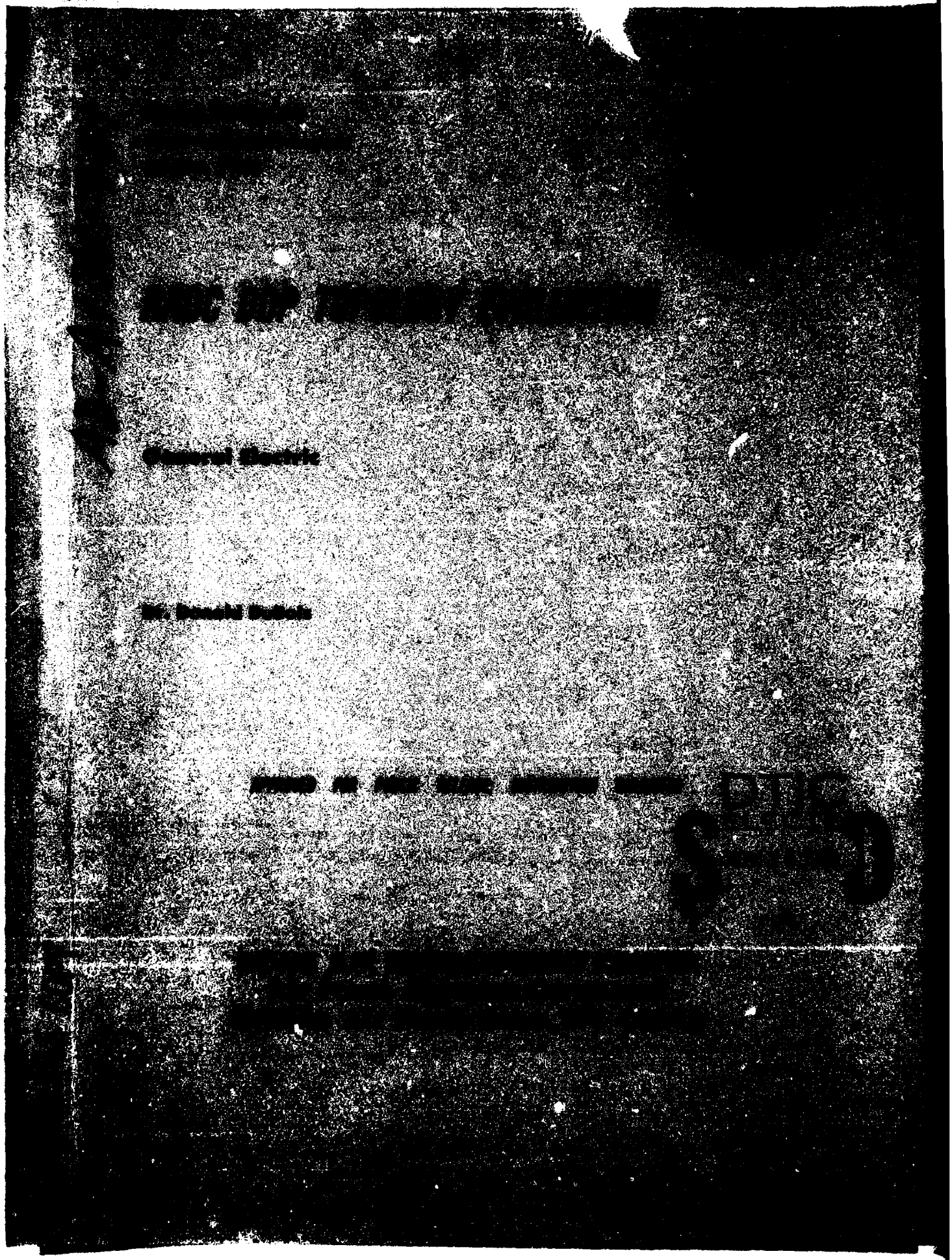

AD-A134 701 RADC (ROME AIR DEVELOPMENT CENTER) DDP (DISTRIBUTED DATA PROCESSING) TOPOLOGY EVALUATION(U) GENERAL ELECTRIC CO SUNNYVALE CALIF D DUBOIS AUG 83
1/2
UNCLASSIFIED RADC-TR-83-79 F30602-80-C-0267 F/G 9/2 NL




MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



[redacted]

[redacted]
Acting Chief, Command and Control Division

FOR THE COMMANDER:

[redacted]
Acting Chief, Plans

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER<br>RADC-TR-83-79 | 2. GOVT ACCESSION NO.<br>AD-A134 | 3. RECIPIENT'S CATALOG NUMBER<br>701 |
| 4. TITLE (and Subtitle)<br>RADC DDP TOPOLOGY EVALUATION | | 5. TYPE OF REPORT & PERIOD COVERED<br>Final Technical Report<br>25 Aug 80 - 25 Sep 82 |
| | | 6. PERFORMING ORG. REPORT NUMBER<br>N/A |
| 7. AUTHOR(s)<br>Dr. Donald DuBois | | 8. CONTRACT OR GRANT NUMBER(s)<br>F30602-80-C-0267 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>General Electric<br>Space Systems Division<br>1277 Orleans Drive, Sunnyvale CA 94086 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS<br>62702F<br>55812125 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Rome Air Development Center (COTD)<br>Griffiss AFB NY 13441 | | 12. REPORT DATE<br>August 1983 |
| | | 13. NUMBER OF PAGES<br>186 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)<br>Same | | 15. SECURITY CLASS. (of this report)<br>UNCLASSIFIED |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>N/A |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Patricia J. Baskinger (COTD)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Modeling
Simulation
Distributed Systems
Networking

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report presents a technical overview of the Distributed System Simulator (DSS), a modeling tool which can be used to facilitate the performance analysis of computer systems and networks through simulation. It is implemented in Simscript II.5 (an event oriented simulation language) and ECSS II (a special-purpose simulation language designed to simulate computer systems). The DSS models may use any of the statements from these two languages; its major function is to extend the capability

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

of these languages to model computer networks.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

LIST OF TABLES

# SECTION 1
# INTRODUCTION

## 1.1 TASK OVERVIEW

The Distributed Data Processing Topology Evaluation Project is being conducted in support of the Rome Air Development Center's (RADC) mission to provide standards and technical guidance in the effort to develop and implement distributed data processing networks.

## 1.2 TASK OBJECTIVES

The purpose of this project is to develop and install on the RADC HIS 6180 computer facility a user-oriented modeling capability for computer networks. This capability should be easy to learn and use, provide an easily readable language for model description, minimize the time and effort required for model development, produce easily modified models, produce models that are economical to run and produce clear and concise tabular output statistics.

This project had two major phases. In the first phase a simulator development tool called the Distributed System Simulator (DSS) has been developed along with three high level models of computer networks. These models have the ability to yield performance measures to evaluate DDP configurations for given workloads. These models include, as a minimum, nodes for various computers, varying processing loads and rates for each node, and different band widths of communication lines between nodes.

During the second phase, DSS has been used to develop three detailed models of computer networks using the ISO Reference Model as a framework. These detailed models include a communication protocol (CP) model, a reliability/availability (R/A) model and a distributed data base (DB) model. These models follow the ISO architecture framework in that each succeeding model uses the services of the preceeding model in a hierarchical fashion. The communication protocol model simulates the X.25 interface for packet switched networks (levels one through three of the ISO Reference Model). Adaptive routing procedures necessitated by nodal failures are simulated in the reliability/availability model.

Finally DSS was used to build a distributed data base model (the application level, layer 7, of the ISO reference model) using the facilities provided by the CP and R/A models.

## 1.3 THE DISTRIBUTED SYSTEM SIMULATOR (DSS)

The Distributed System Simulator (DSS) has been developed as a modeling tool to facilitate the performance analysis of computer networks through discrete event simulations. There are two broad categories of problems that had to be addressed in designing DSS. The first is the large set of problems which naturally arise in providing a simulation tool that would be applicable to many types of computer networks. Even a brief survey of the literature describing computer networks and the likely developments in the future strongly suggest that there are no typical networks. The design of computer networks is in its infancy and therefore constantly evolving on both the software and hardware levels.

The second broad problem that the design of DSS has addressed is the fact that simulation, expecially of diverse and complex systems such as computer networks, can be a time consuming and costly exercise. DSS has been designed to minimize the development time of simulators and to aid in their debugging and verification phases. DSS is a precompiler which has as a subset a language specifically designed for simulating single and multiple processing systems called ECSS (Extendable Computer System Simulator) [DOSY 75]. The output of DSS is an ECSS program which is translated into Simscript II.5, compiled and run. DSS has been designed as a modelling tool which has special facilities for simulating a broad range of computer networks.

## 1.4 BACKGROUND

A survey of research efforts on the simulation of computer networks reveals two major parellel trends. The first is the development of special-purpose high-level languages to simulate computer systems. These include, among others ASPOL [MACD 13], CSS (an IBM product) [SEAM 69], and ECSS II (Extendable Computer System Simulator) [KOSY 75, UNGE 78]. Generally these

languages have evolved from high-level languages such as FORTRAN and ALGOL 68 or general-purpose simulation languages such as Simscript II.5 [KIVI 73]. Simulation can be a time consuming and costly exercise. Special-purpose simulation languages help to alleviate this problem. They make the specification of computer systems easier. Trade facilities and detailed output reports reduce the time needed to verify and validate a simulator. A typical problem with such languages is that increased specialization implies decreased flexibility. ECSS II, along with some other languages, has solved this problem by allowing all valid Simscript II.5 statements in an ECSS simulator. This is an important factor in the usefulness of these languages since high-level constructs, which reduce the time needed to build most simulators, cannot be expected to meet all model's needs. The next lower level, Simscript II.5 in the case of DSS, provides complete flexibility (Figure 1.4-1).

The second major trend is that some of these simulators have increased applicability with respect to the range of the systems they can model. At one end of the spectrum are (see Figure 1.4-2) case studies, specially designed simulators to study particular problems of computer systems [MACD 67, HUTC 73]. Usually these simulators are not used once the modeling project for which they were designed has ended. Simulators with a wider scope of applicability are parametric models [PRIC 77, SHOE 78]. With these simulators it is possible to vary certain well-defined parameters such as the input rate of application jobs and average message lengths. By varying the input parameters, it is possible to perform a range of experiments. However, these simulators are restricted to very specific assumptions about the system architecture being modeled. Further, the user cannot change the network protocols, such as routing algorithms and flow control procedures. A simulator that goes beyond the limitation of parametric models, called a structural simulation model, has been developed [SCHN 78]. This system, named VANS (Value Added Network Simulator), assumes that the communications network is a store-and-forward packet-switched network like the ARPANET. Instead of having fixed protocols, the user may replace a subprogram that is currently

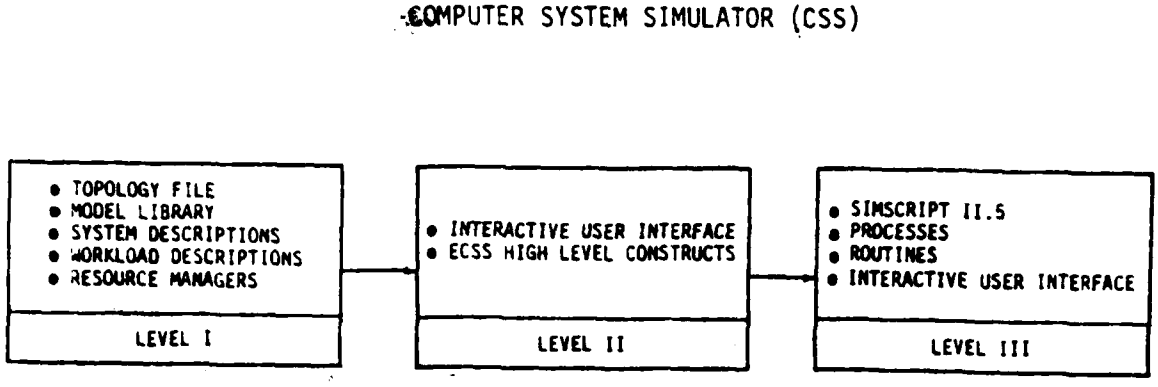



Figure 1.4-1 DSS Model Description Levels

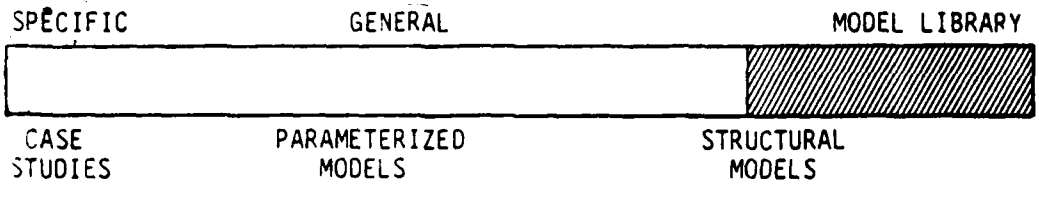



Figure 1.4-2 Range of Model Generality - Case Studies to Model Libraries

# SECTION 2
# OVERVIEW OF DSS

DSS has been designed as a modeling tool to facilitate the performance analysis of computer systems and networks through simulation. It is implemented in Simscript II.5 (an event oriented simulation language) and ECSS II (a special-purpose simulation language designed to simulate computer systems). The DSS models may use any of the statements from these two languages; its major function is to extend the capability of these languages to model computer networks.

A separate DSS model can simulate a single node in a computer network. (A node is simply a collection of system components that communicate with other components of the network over some transmission medium. See Section 2.1 for a fuller description.) If two or more nodes have similar characteristics, DSS can duplicate the model as many times as there are similar nodes. This capability greatly reduces the amount of code to be generated by the user. These DSS models are then combined to form a simulator of the entire network. DSS models are not limited to simulating a specific architecture or set of protocols since they have access to all of the high-level constructs of ECSS II and Simscript II.5. There are several advantages to having a separate model for each node. First, DSS provides the capability of debugging and verifying DSS models separately. In a network with fifty nodes, there may be only two distinct DSS models: one for the switching computers and one for the host sites. Instead of trying to verify a fifty-node network simulator, the problem is reduced to verifying two DSS models separately. The second advantage is that a library of DSS models may be created that can focus on particular network problems, such as flow control or routing algorithms. These DSS models may be used again in other simulators so that, as the library grows, the time to build a simulator can be reduced in some cases. In this sense DSS is an extendable system.

DSS has an internodal communication facility that is the mechanism by which one DSS model communicates with another. This facility allows messages of

arbitrary size and content to be passed between models without presuppositions concerning the communication protocols that are simulated. In other words, the DSS communication facility is the means by which separate DSS models are connected to form a simulator of an entire network without biasing the user to favor one communication protocol over another.

There are three main input files to DSS that a user must supply. The first is the file consisting of the DSS models. These models, described in Section 2.3 may have been previously defined or created by the user. The second input file is the topology file (TP.FILE). This file describes the internodal paths connecting the nodes in the network. The third file (M.FILE) specifies for each node or subsystem in the network the particular DSS model that will simulate it. The user creates these files at a terminal. The DSS is a precompiler that has, as a subset, a language (ECSS) specially designed for simulating single and multiple processing systems. The output of DSS is an ECSS program that is translated into Simscript II.5, compiled and run (Figure 2-1).

## 2.1 NODAL MODELS

DSS views a computer network as a set of interconnected nodes that communicate with each other over some transmission medium. At each of these nodes reside computer resources, such as hardware devices, protocol handlers and data bases. This view of computer networks is very general, so that a broad range of networks from local area networks to geographically dispersed packet switched networks may be modeled.

The user describes each node separately. Models for nodes with similar characteristics may be duplicated automatically. The total network model is a collection of these separate nodal models. The degree of detail in each of these nodal models determines the level of detail of the entire network model. For example, a work station in a local area network that communicates with other stations over a shared bus may be viewed as a separate node. In a satellite/terrestrial system, the satellite and earth stations may be modeled as individual nodes. On a more global level, an entire local area network may

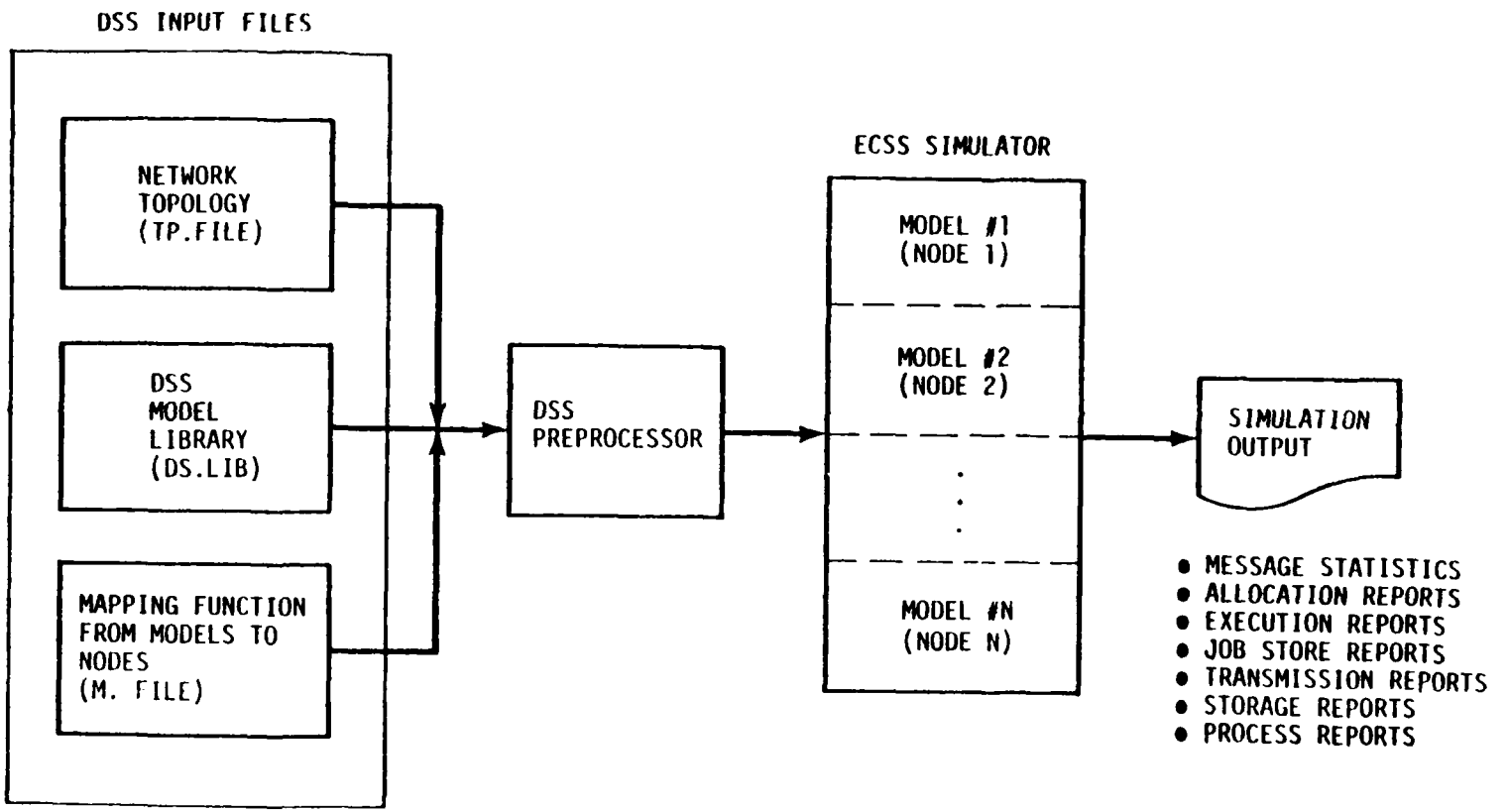



Figure 2-1 Overview of DSS

be modeled as a separate node that communicates with other networks through gateways. The nodal models are stored in a model library to be used as needed. There may be both separate user libraries and a system wide library that all users may access.

Each nodal model is an ECSS program.

DSS extends the capabilities of ECSS in two important ways:

- A library of models may be developed that can be used in more than one simulator.
- Nodal models can be duplicated any number of times for similar subsystems in the overall network.

This allows DSS to be an extendable system and minimizes the time needed to build simulators. To provide these capabilities, DSS is a preprocessor of ECSS programs. ECSS output is translated into Simscript, compiled, and run. Models may reside in the model library as compiled programs, reducing initial configuration time.

Regardless of the detail level of a node, each node may be modeled as the interaction of three main components: a set of resources with finite capacity, (System Description); the tasks or demands the system is designed to service (Workload Description); and the allocation policies that determine how these resources are to be apportioned among the jobs (Resource Manager Description). The next three subsections describe each of these components. Section 2.2.5 gives an example of a complete nodal model using the System Description, Workload Description, and Resource Manager sections.

## 2.2 EXTENDABLE COMPUTER SYSTEM SIMULATOR (ECSS)

ECSS is a high level simulation language especially designed to simulate single or multiple processing systems. ECSS contains as a subset Simscript II.5 which is a general purpose simulation language [KIVI 73]. The main purpose of this section is to highlight major parts of ECSS so that the following sections of this report may be more easily understood. A more comprehensive view of ECSS is contained in [KOSY 75].

### 2.2.1 SYSTEM DESCRIPTION

The System Description section defines the system resources in terms of hardware devices, their characteristics and interconnections. There are five basic types of devices, each with its own set of properties.

- Private devices
- Storage devices
- Processor devices
- I/O devices
- Job store devices

The name, number of members, and characteristics of a class of devices are declared by a SPECIFY statement:

SPECIFY I Device Class Name EACH
Characteristic Clauses

An example of a SPECIFY statement:

```
SPECIFY 2 PROCESSORS EACH
   EXECUTES 500000 INSTRUCTIONS/SEC,
   STORES JOBS FOR EXECUTION,
   TRANSMITS 50000 BYTES/SEC
```

In this statement two processors are defined. Each one has three specific properties defined by the characteristic clauses. As this example illustrates, a device may have properties from more than one of the five basic types of devices. 'I' is the number of devices being defined in the SPECIFY statement. 'Device Class Name' is an arbitrary variable name, and a 'Characteristic Clause' defines a property from one of the five basic types of devices. There is a total of 10 possible characteristic clauses that describe the properties of the basic device types.

The interconnections between devices are defined in the System Description Section by means of PATH statements with the general form:

```
PATH      name      CONNECTS
   Device Name 1   TO
   Device Name 2   TO
   :
   Device Name N
```

For example, assuming that devices DISK and CHANNEL were defined by SPECIFY statements, they may be interconnected by the PATH statement:

```
PATH IO CONNECTS
   DISK TO
   CHANNEL
```

The System Description section, as can be seen from these examples, is a high level, self documenting language.

### 2.2.2 WORKLOAD DESCRIPTION

The second major part of a nodal model is the Workload Description section. This is where the load on the resources of a computer system is described. The basic component of this section is a simulation process that may be described as a sequence of related events. For example, the pattern of resource requests that comprise a computer job may be defined as one process.

A job is initiated on a particular processor by the statement:

```
START A      Job Name      ON      Device Name      WITH PRIORITY E
```

'Job Name' is the unique identifying name for the job; 'Device Name' is the predefined processor device on which the job is to run; 'E' is the priority of the job, i.e., higher priority jobs interrupt lower priority jobs on the processor (this may be varied as described in the Resource Manager Section).

Two examples of the basic statements that simulate resource requests of computer jobs are as follows.

Example:

```
EXECUTE  N INSTRUCTIONS
```

The 'EXECUTE' statement holds a processing device for a time that depends on the instruction rate of the processor and the number of instructions in the statement. The processor is the device upon which runs the job that was declared in the START statement at activation time.

Example:

```
SEND  N DATA.UNITS  VIA  PATH NAME
```

The SEND statement causes a set of I/O devices to be held for a simulated time. This set is specified implicitly by the 'PATH NAME' that had previously been defined in the System Description section and that logically connects a string of devices such as channels and disks. The length of the simulated time depends on the transmission speed of the slowest device at either end of the path and the number of DATA.UNITS in the SEND statement.

### 2.2.3 RESOURCE MANAGER DESCRIPTIONS

The allocation policies for the resources of a node depend on the operating system. This component of real systems is simulated by resource managers. There is a resource manager for each of the five basic types of devices. For example, when a job runs on a particular processor, the order in which it will be served (e.g., round robin or pre-emptive priority) is determined by the Execution Manager. EXECUTE statements in jobs invoke the Execution Manager. Resource managers are ECSS/Simscript routines to which the user has access and which he may alter. This factor gives DSS great flexibility in the types of operating systems that may be simulated.

By dividing a nodal model into the three categories of System Description, Workload Description and Resource Managers, it is possible to run one Workload Description section using the System Description section of another simulator (providing the names of the devices are compatible). This feature is quite valuable when performing simulation experiments for different computer design alternatives. The amount of simulator effort is greatly reduced.

### 2.2.4 SIMSCRIPT II.5

As stated above ECSS is a superset of Simscript II.5. Any Simscript statement is allowed in an ECSS program. Simscript II.5 is a full programming language designed specifically for simulation. Simscript II.5 provides a general purpose high-level base language, comparable in power with PL/1 and ALGOL. The base language is augmented with the facilities necessary for simulation:

- The entity-attribute-set "world view" of Simscript
- Both internal and external events
- Process and resource orientation
- A large collection of random number distribution generation procedures
  - Beta
  - Binomial
  - Erlang
  - Exponential
  - Gamma
  - Hyperexponential (DSS)
  - Hypoexponential (DSS)
  - Log Normal
  - Normal
  - Poisson
  - Uniform (Discrete)
  - Uniform (Continuous)
  - Weibull
- Automatic statistics collection, triggered by the nonprocedural ACCUMULATE and TALLY operations
- The report generator of Simscript

Simscript II.5 is implemented to handle large simulation models. Virtual storage is provided for all data structures.

### 2.2.5 AN EXAMPLE OF A COMPLETE NODAL MODEL

In this section we briefly describe how the various statements from the System Description, Workload Description and Resource Manager Sections can be combined to form a complete nodal model. The model will represent an on-line text-editor system. This model may be saved in the model library and used as a stand-alone model or combined with other models from the library to form a larger network-wide model. (This example is taken from the ECSS User's Manual [KOSY 75]).

The configuration for this hypothetical system being modeled is depicted in Figure 2.2.5-1. It consists of a central computer, two channels, three random-access disks, 20 terminals, and a printer. Although one of these elements does contain a processor, there are also I/O devices, data paths, storage and other types of resources. This hardware is to be used for on-line text editing by up to 20 users simultaneously. The System Description section and the Workload Description section which model this configuration are shown in Figure 2.2.5-2. Lines 1 through 30 of this figure constitute the System Description section. At the top of this Section, a device called CPU is specified to have three components: a job store, a processor, and a storage component. Since all users employ the same text-editor program, the system has been designed such that enough main memory has been set aside to contain it, including a block of working storage for each user. We are not interested in this portion of memory in this model, thus it is not represented. We are interested in the remaining memory which is divided into ten message buffers that can be allocated dynamically to users as necessary, and four batch partitions. We assume that each buffer and partition is identical to the others, so its actual size does not matter in the model. The time-unit for this model is a second; and so the CPU's average execution rate is 100,000 instructions/ second.

The twenty TERMINALS are private I/O devices having a 150 character/second transmission rate. They are private because we will want to allocate one to each user exclusively.

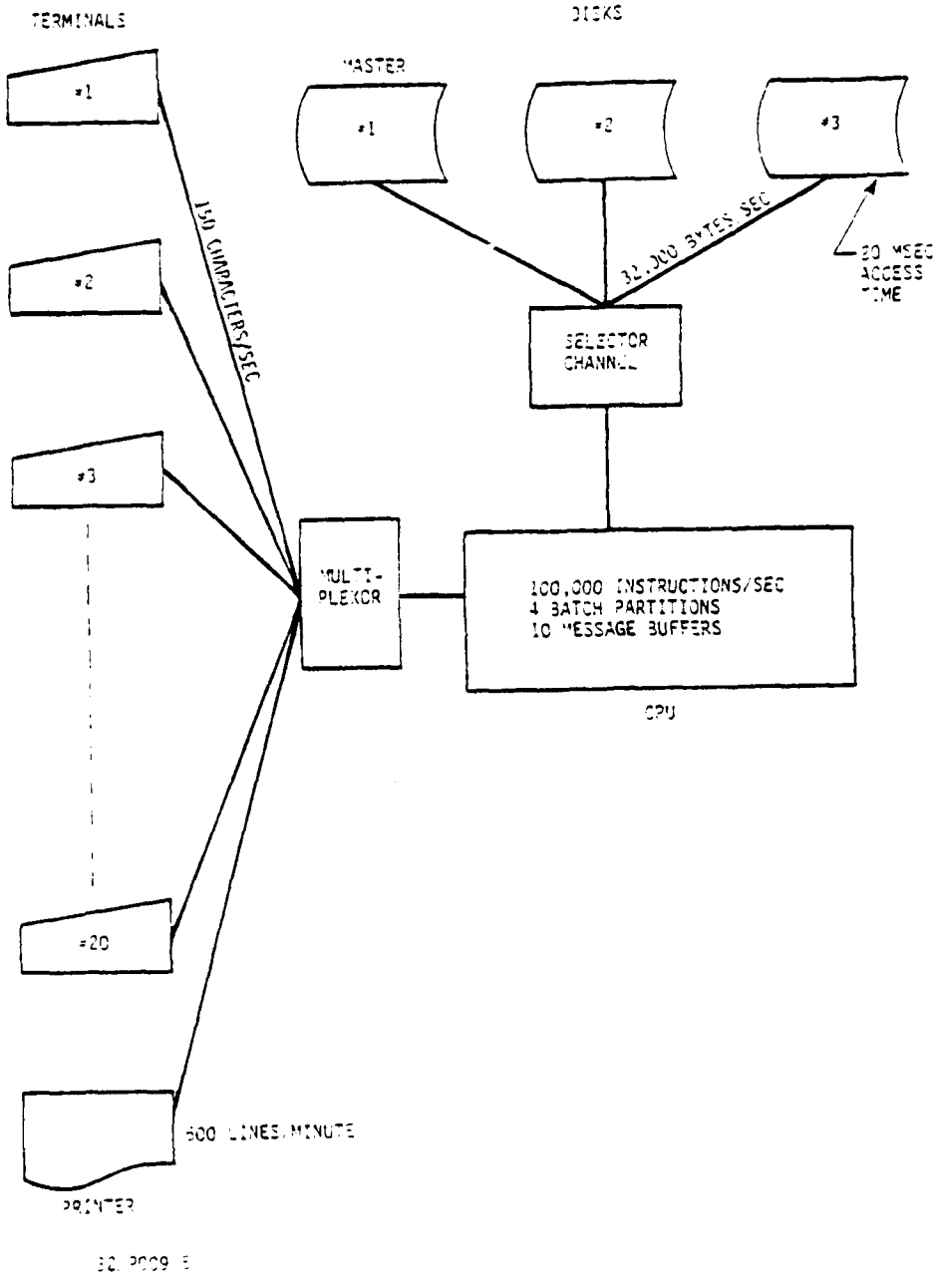



Figure 2.2.5-1 System Configuration for a On-line Test Editor

CH1 represents a 3-port multiplexor channel and CH2 represents a selector channel. Since no transmission rates have been specified, messages between them will proceed at the speed of the devices to which they connect.

The three DISKS are I/O devices which can transfer data at 32,000 bytes/second. They are fixed head disks with a rotational period of 40 milliseconds and contain 2 records per track. The average latency is 20 milliseconds. Text files for the various users are stored on each disk.

One disk, designated as the MASTER.DISK, also stores a file used to check the validity of a user's account number at log on, and to record system utilization data for billing purposes. When this file is in use, it is protected from tampering by using the ACCT.FILE private device as a software lock.

The PRINTER is a simple I/O device representing a 600 line per minute printer. The last lines in the system description indicate the data paths connecting the channels to each of the other I/O devices. Although the CPU will not participate in simulated I/O operations, its logical connection to the channels is indicated in the CONNECTS clause.

As in any model, this is a simplification of a real system. For example, no consideration has been given to disk or terminal control units, though these could be easily added if considered important to system performance. Using an average latency is another simplification which could be modified by supplying a latency function. In general, System Descriptions are written to include only the most important resources, described at a level of detail consistent with the rest of the model. This example is not meant to be a complete presentation of the model description language, but serves only to give an indication of its self-documenting nature and the wide range of resources that may be defined. Overhead clauses are not necessary and may be left out of a System Description section when the level of detail permits it.

```
1.  SYSTEM DESCRIPTION "The ON-LINE TEXT-EDITOR SYSTEM
2.
3.     SPECIFY 1 CPU WHICH
4.        STORES UP TO 4 BATCH JOBS,
5.        EXECUTES 100000 INSTRUCTIONS/TIME.UNIT,
6.        HAS CAPACITY OF 30 BUFFERS, AND
7.        CONNECT TO (CH1,CH2)
8.
9.     SPECIFY 20 PRIVATE TERMINALS,
10.       EACH TRANSMITS 150 CHARACTERS/TIME.UNIT
11.
12.    SPECIFY 1 CHI WHICH TRANSFERS 3 MESSAGES CONCURRENTLY
13.
14.    SPECIFY 1 MULTIPLEXER WHICH TRANSFERS 1 MESSAGE
15.
16.    SPECIFY 2 DISKS, EACH
17.       TRANSMITS 30000 BYTES/TIME.UNIT, AND
18.       ABSORBS 0.0 TIME.UNITS PER MESSAGE
19.
20.    IDENTIFY DISKS@1 AS MASTER.DISK
21.
22.    SPECIFY 1 PRIVATE ACCTG FILE
23.
24.    SPECIFY 1 PRINTER WHICH TRANSMITS 10 LINES/TIME.UNIT
25.
26.    PATH 1 PATH CONNECTS THE 20 TERMINALS
27.    PATH DISK.PATH CONNECTS CH2 TO DISKS
28.    PATH LP PATH CONNECTS CH1 TO PRINTER
29.    THE MULTIPROGRAMMING MANAGER IS WORK.POOL
30. "END"
31.
32. "EXTERNAL DESCRIPTIONS"
33.
34. EXTERNAL PROCESS USER
35.
36.    ALLOCATE FROM TERMINALS AS CRT  "SIGN ON"
37.    GET VOLUME=RANDOM(1,20,1),NUMBER=1,1,1 "GET USER ID AND VOLUME"
38.    GET PASSWORD "CHANGE IF DESIRED"
39.    START TEXT.EDITOR "ACCOUNT" GIVEN CRT,VOLUME,1,0
40.       "PASSWORD IF ANY"  "PRIORITY"
41.    WAIT FOR SIGNAL                "AWAIT ACKNOWLEDGEMENT"
42.
43.       IF 1 "PROBABILITY" 0.1 THEN
44.
45.          SEND 11 CHARACTERS FROM DEVICE(CRT) TO CHI VIA T.PATH
46.                                 "AND WAITING FOR A RESPONSE MESSAGE"
47.
48.          WAIT UP TO 30 TIME.UNITS
49.
50.       LOOP
51.
52.    SUBMIT LISTOFF GIVEN VOLUME, EXPONENTIAL(1,200.0,1) TO CPU
53.    DEALLOCATE THIS CRT
54.
55. END "EXTERNAL PROCESS USER
56.
57. JOB TEXT.EDITOR GIVEN USER, VOL, INTERACTIONS
58.
59.    ALLOCATE ACCTG.FILE              "LOGIN PROCESSING
60.    RECEIVE 500 BYTES FROM MASTER.DISK VIA DISK.PATH WITH PRIORITY
61.    DEALLOCATE ACCTG.FILE
62.    EXECUTE 3500 INSTRUCTIONS
63.    RECEIVE 2200 BYTES FROM DISK@VOL VIA DISK.PATH WITH PRIORITY 1
64.    SIGNAL USER
65.
66.
67.    WHILE INTERACTIONS > 0 DO
68.
69.       WAIT FOR INPUT
70.
71.       EXECUTE EXPONENTIAL(1,2500.0,1) INSTRUCTIONS
72.       GET RANDOM(1,1,4,
73.       IF 1 = 1, GET 1-4 BUFFERS FROM CPU REGARDLESS
74.       SEND 1 * 40 CHARACTERS VIA T.PATH AS A RESPONSE
75.          WAITING FOR COMPLETION AT LINE
76.
77.       IF RANDOM(1,1,1) = 1              PLATE REQUIRED
78.          SEND 3500 BYTES TO DISK@VOL VIA DISK.PATH
79.          EXECUTE 4250 INSTRUCTIONS
80.          REGARDLESS
81.
82.    "LINE"
83.       IF 1 = 1, FREE 1-4 BUFFERS ON HAND REGARDLESS
84.
85.       SUBTRACT 1 FROM INTERACTIONS
86.       LOOP
87.
88.    EXECUTE 500 INSTRUCTIONS          "SIGN OFF"
89.    ALLOCATE ACCTG.FILE
90.    RECEIVE 500 BYTES FROM MASTER.DISK VIA DISK.PATH
91.    SEND 500 BYTES TO MASTER.DISK VIA DISK.PATH
92.    DEALLOCATE ACCTG.FILE
93.
94. END "TEXT.EDITOR JOB
95. JOB LISTOFF GIVEN VOL, ...
96.    DEFINE LNS AS A REAL VARIABLE
97.    RECEIVE LNS*80 BYTES FROM DISK@VOL VIA DISK.PATH
98.    SEND LNS LINES VIA LP.PATH
99. END "LIST OFF" JOB
100. END "WORKLOAD
```

Figure 2.2.5-2 A Complete Simulation Model

Line 29 declares that the CPU is allocated to jobs on a round robin basis. The default manager for processor devices is first-come-first-served. Changing this one line in effect changes the operating system model (that is, the resource manager for the CPU) without having to change any other lines in the model. There is a library of resource managers available to the user. Alternatively, the user may choose to write his own resource manager for a particular device or class of devices. This new manager would be given a unique name and stored in a system wide or user-defined library for resource managers. We now turn to the Workload Description section.

For this example, each user at his terminal makes a related series of demands on the system, which cause different kinds of internal operations and requires rapid, interactive responses. The applications programs driving the system must produce these responses and perform the operations. This activity includes logging users in and out, transferring text data from disk storage to main memory and back, performing the editing operations, and printing the results, thus utilizing processor, storage, I/O and other resources as necessary for each function. Using a model, it is possible to determine the average time it takes to respond to a user request, and to evaluate the effect of different numbers of users on response time, the utilization of each device, the effect of queueing delays, to determine how the system performs under various conditions, and to indentify the important factors in determining that performance.

The Workload Description section for this hypothetical system is described in lines 32 through 100 of Figure 2.2.5-2. It contains three processes: one external process and two jobs.

- The External Process USER

  Lines 34 through 55 describe the behavior of each user. The statements represent the events and activities charted in Figure 2.2.5-3 which is a scenario of text editing behavior.

- The TEXT.EDITOR Job

  The TEXT.EDITOR job defined by lines 57 through 94 is more complex than the user process. It contains more activities and uses more resources. All jobs in this model are executed by the CPU processor concurrently, i.e., the processor is multiprogrammed.

- The LISTOFF Job

  The LISTOFF job described by lines 95 through 99 represents data transfer from a text file to main memory and then to the printer. The amount, (LNS), is interpreted as the number of lines averaging 80 bytes per line. The text file is specified by the number of the disk on which it is stored. A single message represents the entire printing activity; consequently, there is no danger that segments of different listings will be interspersed by LISTOFF jobs competing for the printer.

Jobs may be parameterized so as to have general processing characteristics. Each job, under its own name, may be saved in a Workload Description library that is user specific or available on a system-wide basis. For instance, there may be a graphics output job called JOB GRAPHICS that requires as parameters (which may be defined as a menu of options) the format and data rate of the graphical output to be modeled. This job would then simulate the CPU, channel, and disk overhead that would be incurred by a real graphics display terminal.

It should be noted that each major component of a nodal model (the System Description, Workload Description, and Resource Manager sections) allows for the construction of models in a modular fashion. Changes in one component do not necessitate changes to another component. In this brief example, the workload could be redefined and scientific application jobs run against the same System Description and Resource Manager sections. This modularity facilitates the running of model experiments as system components are varied.

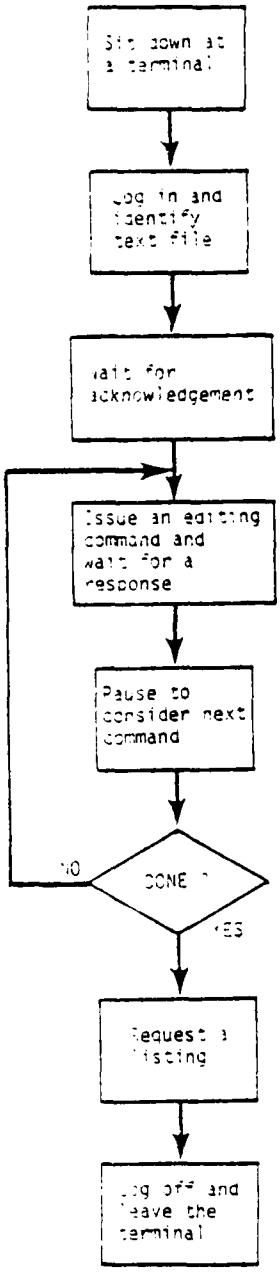



Figure 2.2.5-3 Test-Editor User Behavior

## 2.3 DSS MODELS

As stated above a DSS model is essentially an ECSS program that models the activity within a single node of a network. The user may design different DSS Models independently and later connect the models to form a simulator for a network. This capability to simulate a node with an individual DSS model encourages a modular approach to the design of network simulators. Also, modifications within a single node can be easily made without changing the rest of the network model. Finally, as the user develops new DSS Models they may be stored in a DSS Model Library. The models can be retrieved and used for future simulations.

### 2.3.1 DSS MODEL DEFINITION

Since each DSS model conforms to ECSS syntax it may contain its own Preamble, System and Workload Description sections, as well as Simscript routines. Each DSS model must be bracketed by a model number statement (eg., Model #1) and an END statement. Table 2.3.1-1 is a schematic representation of the basic structure of a DSS model. An entire program file consisting of several DSS models is outlined in Table 2.3.1-2. Generally a DSS simulation consists of more than one model from the DSS model library. Figure 2.3.1-2 is an actual example of a DSS simulation program.

Table 2.3.1-1 Standard DSS Model Program File

```
MODEL #1
    PREAMBLE [optional]
        .
        .                                   Preamble  (optional)
        .
    END "PREAMBLE
    SYSTEM DESCRIPTION
        .
        .                                   System Description
        .
    END "SYSTEM DESCRIPTION
    WORKLOAD DESCRIPTION
        .
        .                                   Workload Description  (optional)
        .
    END "WORKLOAD DESCRIPTION
    SIMSCRIPT II.5 ROUTINES
        .
        .                                   Routines  (optional)
        .
    END "MODEL #1
```

Figure 2.3.1-1 DSS Model Structure

```
PREAMBLE "SYSTEM WIDE PREAMBLE
.
.                                           NETWORK WIDE
.                                           PREAMBLE
END "PREAMBLE
MODEL #X
    SYSTEM DESCRIPTION
    .
    .
    WORKLOAD DESCRIPTION                    CODE FOR MODEL X
    .
    .
    ROUTINES
END "MODEL #X
.
.
.
MODEL #Y
    SYSTEM DESCRIPTION
    .
    .
    .                                       CODE FOR MODEL Y
    WORKLOAD
    .
    .
    ROUTINES
END "MODEL #Y
```

Table 2.3.1-2 DSS Model Program File Sample

```
PREAMBLE
     EVENT NOTICES
      EVERY MESSAGE HAS AN IO
     DEFINE CLASS AS AN INTEGER VARIABLE
END "NETWORK PREAMBLE

MODEL #1
  PREAMBLE
     PERMANENT ENTITLES
      EVERY ..NODE OWNS AN OVERFLOW
     TEMPORARY ENTITLES
      EVERY ..MSG MAY BELONG TO AN OVERFLOW
END
SYSTEM DESCRIPTION
     SPECIFY 1 CPU, WHICH STORES JOBS FOR EXECUTION
                                                EXECUTES    AT    500000
INSTRUCTIONS / SEC

     SPECIFY 2 CHANNEL, WHICH TRANSFERS MESSAGES AT 2000000
                                                  BYTES/SEC

     SPECIFY  10  TERMINAL,  WHICH  TRANSFERS  MESSAGES  AT  9600
BYTES/SEC

     SPECIFY 1 BUF, WHICH HAS CAPACITY OF 10000 DATA.UNITS
     PATH T.PATH CONNECTS CHANNEL TO TERMINAL
     EXTERNAL PROCESSES ARE CP
     JOBS ARE JBR
END "SYSTEM DESCRIPTION
WORKLOAD DESCRIPTION
     EXTERNAL PROCESS CP GIVEN N
      LET CLASS = 1
     HERE START JBR GIVING N
      WAIT FOR SIGNAL
      LET CLASS = 1
      JUMP BACK
     END "CP
     JOB JBR GIVEN N
       SEND 400 BYTES FROM CHANNEL #1 TO TERMINAL #8
       GET 1000 BYTES FROM BUF
       EXECUTE 5000 INSTRUCTIONS
       FREE 1000 BYTES FROM BUF
       SIGNAL ..CP (N)
     END "JBR
END "WORKLOAD DESCRIPTION
ROUTINE GOPHER
     FOR I = 1 TO 1000
      LOOP
      RETURN
END "GOPHER
END "MODEL #1
```

Table 2.3.1-2 DSS Model Program File Sample (Con't).

```
MODEL #2

PREAMBLE
  TEMPORARY ENTITIES
     EVERY ROOM HAS A DOOR
END "PREAMBLE
SYSTEM DESCRIPTION
     SPECIFY 1 CPU, WHICH STORES JOBS FOR EXECUTION,
               EXECUTES AT 500000 INSTRUCTIONS / SEC
     SPECIFY 1 BUF, WHICH HAS CAPACITY OF 10000 DATA.UNITS
     EXTERNAL PROCESSES ARE CP, XYZ
     JOBS ARE AKNOL
END "SYSTEM DESCRIPTION
WORKLOAD DESCRIPTION
     EXTERNAL PROCESS CP GIVEN K
          HERE IF CLASS = 1
               START AKNOL ON CPU
          ALWAYS
          START XYZ
          WAIT 5 SEC
          JUMP BACK
     END "CP
     EXTERNAL PROCESS XYZ
          WAIT 2 SEC
          LET CLASS = 0
     END "XYZ
     JOB AKNOL
          GET 1000 BYTES FROM BUF
          EXECUTE 1000 INSTRUCTIONS
          FREE 1000 BYTES FROM BOF
     END
END "WORKLOAD DESCRIPTION
     ROUTINE HOG LET X = TIME.S + 4.0
     RETURN
     END "HOG
END "MODEL #2
```

### 2.3.2 INTERNODAL COMMUNICATION

Because DSS models are created independently of each other there is a need for a standard mechanism for inter-nodal communication. The communication facility within DSS was designed with this purpose in mind.

In a network with M nodes each of the nodes has an active instance of an external process associated with it (Figure 2.3.2-1). For example, node 2 is associated with external process CP02. The CPXX process (where 'XX' stands for a nodal number from 1 to N) is responsible for all inter-nodal communication. Communication between nodes is simulated by four basic steps.

In describing these steps we shall assume that the communication between two arbitrary nodes in a network, called N1 and N2, is to be simulated, and that there is a direct link, an ECSS path named IO1A.N1.N2 connecting these two nodes.

STEP 1 Simulate the transmission times between nodes N1 and N2 with a standard ECSS SEND statement using path. The form of this statement is:

```
SEND n data units VIA  IO1A.N1.N2
```

where n data units is the length of the message expressed in bits, bytes or some other previously chosen unit of information.

After the delay is simulated actual message data must be passed between nodes N1 and N2. We assume that a message entity (simply an area of memory) has been created with information contained in it such as a source, destination and intermediate nodes traversed. Steps 2 and 3 are responsible for this part of the transmission.

STEP 2 Put a pointer to the message that is being sent from N1 to N2 in the message file of node N2. The code for this step is:

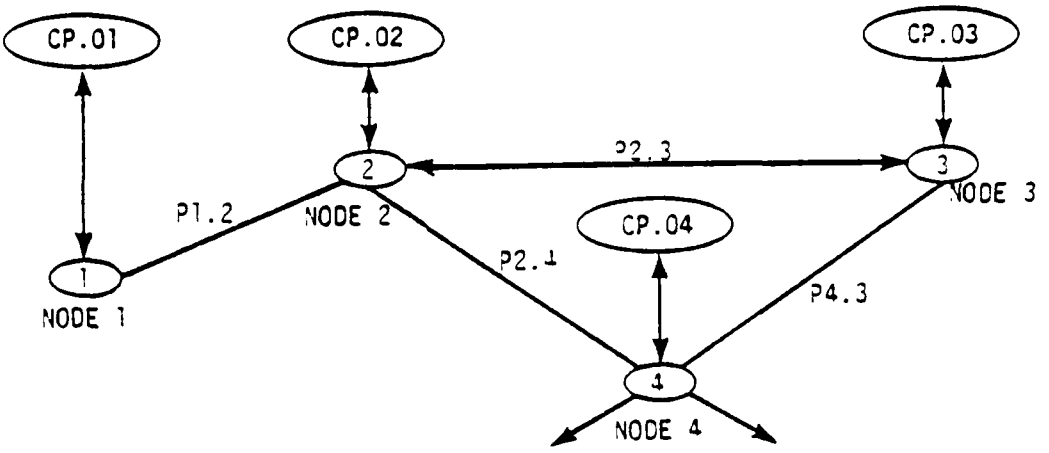



Figure 2.3.2-1 A logical View of Communicating Nodes

FILE MSG IN ..MGFILE (N2)

where MSG is a pointer to the defined temporary entity which is the internodal message; ..MGFILE is a FIFO set owned by the receiving node, in this case node N2. DSS creates a message file called ..MGFILE for each node which can be indexed by node number. In this way each node has its own message file reserved for internodal communication.

STEP 3 Notify the communication process for node N2 (CP.N2) that it has a message waiting for it. The code for this step is the standard ECSS SIGNAL statement:

SIGNAL ..CP (N2)

The variable called ..CP is a one dimensional array, created by DSS, which can be indexed by nodal number. It contains the pointers to the unique communicating processes for all of the nodes.

STEP 4 The receiving process, CP.N2, will wake up, look in its message file, find the pointer to the message and take appropriate action depending on the destination of the message, its priority, type etc. The communication process CP.N2 will be in a wait state when signaled by a job or process at node N1. It CP.N2 then removes the message pointer from its own message file, ..MGFILE(N2), and processes the messages in some way. Typical code for this step would be:

```
WAIT FOR SIGNAL
REMOVE THE FIRST MSG          Code in CP.N2 for accepting
FROM ..MGFILE(N2)             incoming message.
IF PRIORITY (MSG) IS ...
```

These four basic steps form the framework for simulating transmission between neighboring nodes regardless of the type of network which is simulated.

The communicating processes are simply logical constructs whose primary function is to pass data from one node to the next. No simulation time is incremented for any of the actions taken by these processes. A communicating process may start jobs on the processing units for the node with which it is associated but the jobs are the users of computer resources, not the communicating process. The communication process within a DSS Model is always called 'CP' and it has exactly one argument. The first line of this process is therefore:

```
EXTERNAL PROCESS CP (NODE.NUMBER)
```

which agrees with ECSS syntax. DSS will append to the name 'CP' the unique nodal ID number so as to create a unique name (and therefore a unique process, not just a separate instance) for each of the nodes in the simulated network. DSS will also activate each of the communication processes at the start of the simulation and pass to each CP process its unique ID number through its one argument. In this way each CP process will have access to its own ID number. This is the one piece of information DSS provides to each of the communication processes. In making routing decisions it is necessary for a communication process to have a local variable containing its own ID number.

A second major function of the communication process is to coordinate all activities within a node. It is the executive process of a node. It performs the initialization functions and starts the jobs and processes which are in the Workload Description of a given node (Figure 2.3.2-2). If JOB.A wants to communicate with JOB.B in a particular node it must first SIGNAL its own CP process, pass a message to the CP process by the mechanism described above, which the CP process then passes on to JOB.B. JOB.B will communicate with JOB.A in the same way.

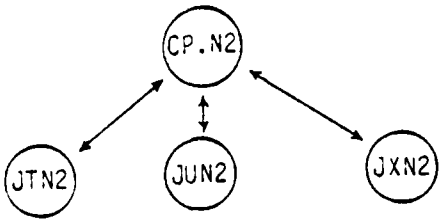



Figure 2.3.2-2
External Process CP

The CP process at each node does not limit the ability to simulate the functions at any node but it does provide a structured way to communicate between nodes and also among jobs and processes within a given node.

The messages in DSS which are passed between nodes have a standard format. Each message is a temporary entity with certain attributes which are defined in Table 2.3.2-3. The TYPE attribute is used in cases where more than one kind of message is transmitted between nodes. For example, a TYPE 1 message might signify a data base update; a TYPE 2 message, a data base query. TYPE 2 messages would require a response from the destination site; a TYPE 1 message might not generate a response and after the update has been completed the message could be destroyed. The communicating process for a node has the responsibility of interpreting the type of messages which arrive at its site and taking the appropriate action - such as starting jobs on processors - based on this and other information contained in the message.

In the three High Level models, message attributes are used in this fashion. In the three detail models, new attributes are defined as required. However, messages (or transactions in the DB Model) are still the media used to activate different functions in the CP Process.

Message Attributes
Table 2.3.2-3

```
TEMPORARY ENTITIES
        EVERY ..MSG HAS AN  ..ID,                                          "
MESSAGE ID
                A   ..TYPE,    " MESSAGE TYPE
                A   ..PT,      " MESSAGE POINTER
                A   ..LGTH,    " MESSAGE LENGTH
                A   ..SRC,     " SOURCE NODE NUMBER OF MESSAGE
                A   ..DEST,    " DESTINATION NODE NUMBER OF MESSAGE
                A   ..Cur,     " CURRENT NOTE NUMBER OF MESSAGE
                A   ..CR.TIME " CREATION TIME OF MESSAAGE
                A   ..NXT.NODE   " NEXT NODE NUMBER MESSAGE IS GOING TO
                A   ..STAT,    " STATISTICAL GROUP OF MESSAGE
                A   ..NPK,     " NUMBER OF PACKETS IN MESSAGE
                A   ..PKID,    " PACKET ID NUMBER
                A   ..T1,      " T1,T2,T3,T4, ARE GENERAL ATTRIBUTES
                A   ..T2,      " OF MESSAGE
                A   ..T3,
                A   ..T4,
                A   ..PRTY,    " MESSAGE PRIORITY
                A   ..R1,      " R1,R2 ARE GENERAL ATTRIBUTES OF
                A   ..R2,      " MESSAGE
                   OWNS A      ..HIST.Q,                                   "NODAL
 HISTORY OF MESSAGE IS CONTAINED
    AND MAY BELONG TO  A      ..MGFILE                                     "  IN
THIS SET

    DEFINE ..CR.TIME,  ..LGTH,  ..R2 AS REAL VARIABLES

        EVERY  ..HST.MSG HAS AN  ..ND,
                    AN  ..ETIME,
            AND BELONGS TO  A   ..HIST.Q
DEFINE  ..ETIME AS REAL VARIABLES
```

Each message owns a set called HIST.Q. When a message enters a node the time of its arrival along with the nodal identification number is saved in the HIST.Q set. This standard procedure allows DSS to trace the route a message takes through the communication network. It is also a device for collecting statistics on messages such as average delay time at a particular node for a given type of message.

The PT attribute of a message is a place for a DSS user to insert a pointer to a temporary or permanent Simscript entity. Even though a message has a fixed format in DSS, this pointer attribute permits complete flexibility in the actual amount of information that is passed from node to node. The entity that is pointed to can be user defined and of arbitrary length. For example, in a very detailed simulation the user defined entity could be the actual header for MESSAGES in a packet switched network. It should be emphasized that when message transmission is simulated only a pointer to a message is placed in the message file of a node since this is the only information that is required in order to have access to the contents of the message.

## 2.4 DSS INPUT FILES DESCRIPTION

In addition to a library of DSS models DSS requires three main input files:

- EXEC File
- TP.FILE
- M.FILE

The content and format for each of these files is discussed in the following three sections.

### 2.4.1 EXEC FILE DESCRIPTION

The EXEC File has two purposes:

1. To provide a means for specifying the dedicated or multiplexed option for interconnection of nodal models;
2. To provide all necessary simulation defaults on the model specific level.

The file is keyword driven and is expandable to meet user requirements. The format of the EXEC File is:

KW OS

where:

KW - is the Keyword
(alpha characters)

OS - option selector
(real or integer value depending on the keyword chosen)

At least one blank between the keyword and the option selector is required.

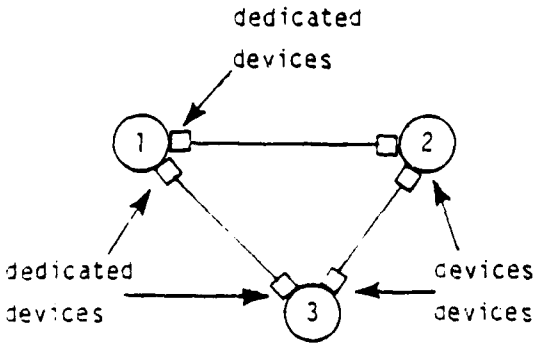
The only keyword in the EXEC File recognized by the DSS preprocessor is "MLT". The value of the option selector for "MLT" - in this case either 0 or 1 - determines whether internodal transmission devices are dedicated or multiplexed. The following section describes this option in greater detail. The other keywords in the EXEC File are model specific and will be described in the sections on the High Level Models (Section 4) or the Detailed Models (Section 5).

2.4.1.1 Multiplexed/Dedicated Option

The difference between the multiplexed and dedicated options is:

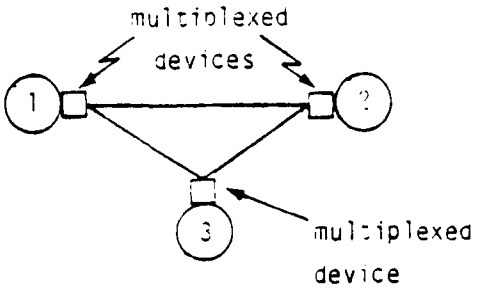
Under the dedicated option, a node will have as many transmission devices as it has paths entering or leaving which connect it to other nodes.

Example of the dedicated option.



A multiplexed device has access to many paths. Under the multiplexed option a node will have only one internodal transmission device regardless of the number of paths.

Example of the multiplexed option.




The user may select this option (dedicated or multiplexed) in three ways:

1. By supplying the EXEC file with a "MLT" as the keyword and a "0" for the option selector, the device configuration will be dedicated.

2. By supplying the EXEC file with a "MLT" as the keyword and a "1" for the option selector, the device configuration will be multiplexed.

3. By omitting the "MLT" keyword and option selector from the EXEC file the device configuration will default to the dedicated option.

Example use of this file is as follows:

MLT 1 - Multiplexed option
MLT 0 - Dedicated option

2.4.2 TP.FILE DESCRIPTION

The TP.FILE's purpose is to allow the user to generate a description of the topological interconnections of the network to be simulated.

The format of the TP.FILE consists of two statement types, the "SPD" and "NL" statements.

The format for the "SPD" statement type is:

```
SPD     S
```

where:

SPD - is the keyword

S - is the transmission device default speed (integer value)

The format for the "NL" statement type is:

```
NL I , S ; J S D ; J S D ; ...
```

where:

NL - is the keyword

I - Key node (integer value - required input)

J - Adjacent node (integer value - required input)

S - Internodal transmission device speed (integer value - optional)

D - Delay associated with the path between node I and J (real value - optional)

The first statement, "SPD", assigns the default speed of the internodal transmission device. This speed is used by the DSS for the devices that are defined in the "NL" statements. A new default speed may be re-entered at any succeeding line in the TP.FILE which will then override the previous default speed.

The second statement, "NL", establishes the connection between the nodes, the internodal transmission device speed at each node and the delay associated with the path between the nodes. Device transmission speeds are optional and will default to the current "SPD" value. Path delays are also optional in the "NL" statement and default to zero.

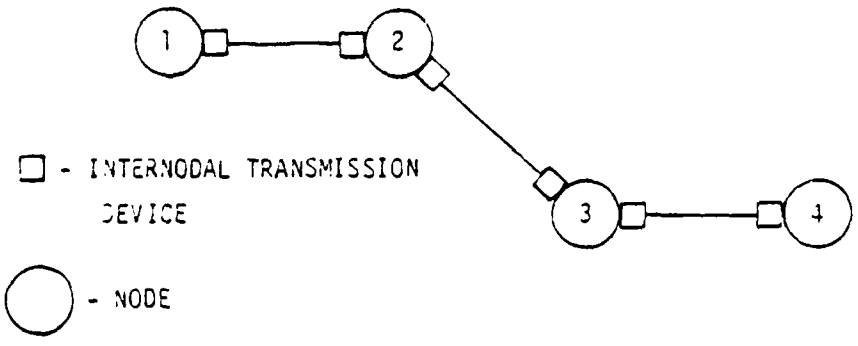
All inputs in the TP.FILE must be separated by at least one blank. Each "NL" statement is limited to 80 characters and cannot be continued on another line. Examples of the use of these two statements follow.

Example 1 - All defaults are in effect.

```
SPD 25000
NL 1 ; 2
NL 3 ; 2 ; 4
```

This example indicates that the transmission device speed is 25000 bytes/sec. The EXEC file "MLT" option is zero; therefore the resulting transmission device configuration is dedicated. The result from this TP.FILE is shown below.

NETWORK TOPOLOGY FOR EXAMPLE 1



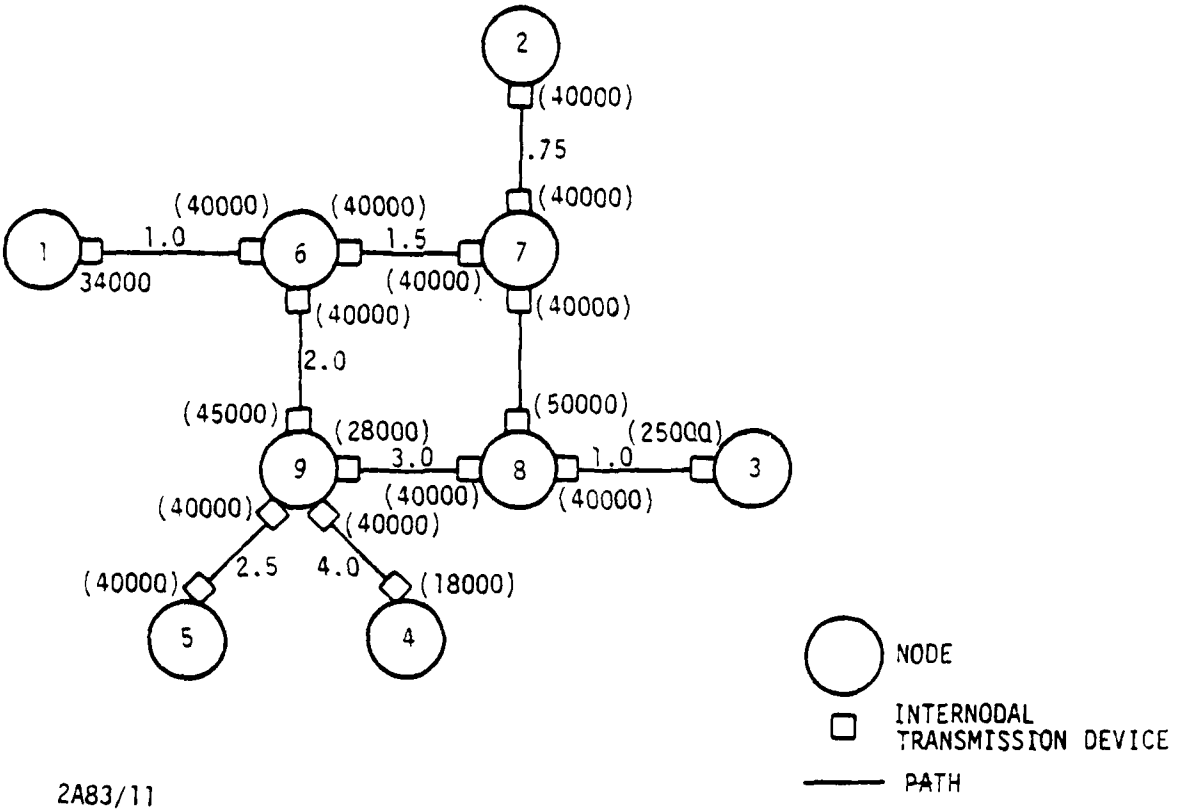

Example 2 - Complex Configuration

```
SPD 40000
NL 6 ; 34000 1.0 ; 7 1. 5 ; 9 45000 2.0
NL 7 ; 2 40000 .75 ; 8 50000
SPD 75000
NL 8 ; 3 25000 1.0 ; 9 28000 3.0
NL 9 ; 4 18000 4.0 ; 5 2.4
```

This example illustrates the explicit assigning of device speeds and path delays.

This example also shows the use of the "SPD" statement to change the transmission device default speed from 40000 to 75000 bytes/sec.

The EXEC file "MLT" option is zero and therefore the resulting transmission device configuration is dedicated. The following figure illustrates this configuration:




2A83/11

2.4.3 M.FILE DESCRIPTION

The function of the M.FILE is to provide the relationship between the node number and the model type associated with that node. The format of the M.FILE is as follows:

```
NN MT
```

where:

NN - node number (Integer value)

MT - model type (Integer value)

At least one blank is required between the node number and the model type. Examples of the use of this file follow.

Example 1 - Four node case.

```
01 01
02 02
03 02
04 01
```

Nodes 1 and 4 are of model type 1; nodes 2 and 3 are of model type 2.

Example 2 - Nine node case.

```
1 01
2 01
3 01
4 01
5 01
6 02
7 02
8 02
9 02
```

This example shows that nodes 1 thru 5 are of model type 1 and nodes 6 thru 9 are of model type 2.

### 2.4.4 DSS INTERNODAL TRANSMISSION DEVICES AND PATHS

DSS offers the user a modular approach to solving network simulation problems. Each node of a network may be simulated with a structured model. The models are joined together to form a simulation model of the entire network.

One of the prime functions of the TP.FILE and M.FILE is to provide the necessary information to the DSS Preprocessor so that these individual models can be joined. DSS uses the information contain in the TP.FILE and M.FILE to create Internodal Transmission Devices and Paths which provide the ECSS link between the individual models.

All Internodal Transmission Device names begin with a "T". Information from the TP.FILE and M.FILE combine to construct the rest of the name. From the TP.FILE the node number is taken. From the M.FILE the associated model type is secured. DSS maintains a count of all Internodal Transmission Devices as they are created for each node. As a device is created, DSS assigns a letter representing its occurence, for example, "A" = First Internodal Device for the node, "B" the second etc.. For each nodal pair (e.g., NL 6 ; 1) two Internodal Transmission Devices, one for each node and an Internodal Path linking the devices are created.

Internodal Paths are created in a similar fashion. All Internodal Paths begin with an "I". Information is combined from the TP.FILE (node number) and the M.FILE (model types of both nodes being connected) to construct the Internodal Path name. DSS also maintains a count of the number of Internodal Paths created at each node.

Suppose the following conditions exist:
The user desires to connect node 06 to node 01. The following TP.FILE statement would accomplish this:

```
NL 6 ; 1    TP.FILE STATEMENT
```

Also the model types for the two nodes are defined in the M.FILE. Suppose node 06 was of model type 2 and node 01 was of model type 1, the M.FILE would contain the following information:

```
01  01      M.FILE
06  02
```

The following illustration (Figure 2.4.4-1) graphically displays the information required to create internodal devices and paths.

In the above example, two Internodal Transmission devices called T06A.M02 and T01A.M01 and an Internodal Path called I06A.N06.N01 are created. The DSS

Preprocessor uses these device and path names to create a SPECIFY statement, and a PATH statement, which connects the transmission devices, in the System Description Section of an ECSS program. For example, the SPECIFY statement that DSS generates for the inter-nodal devices above is:

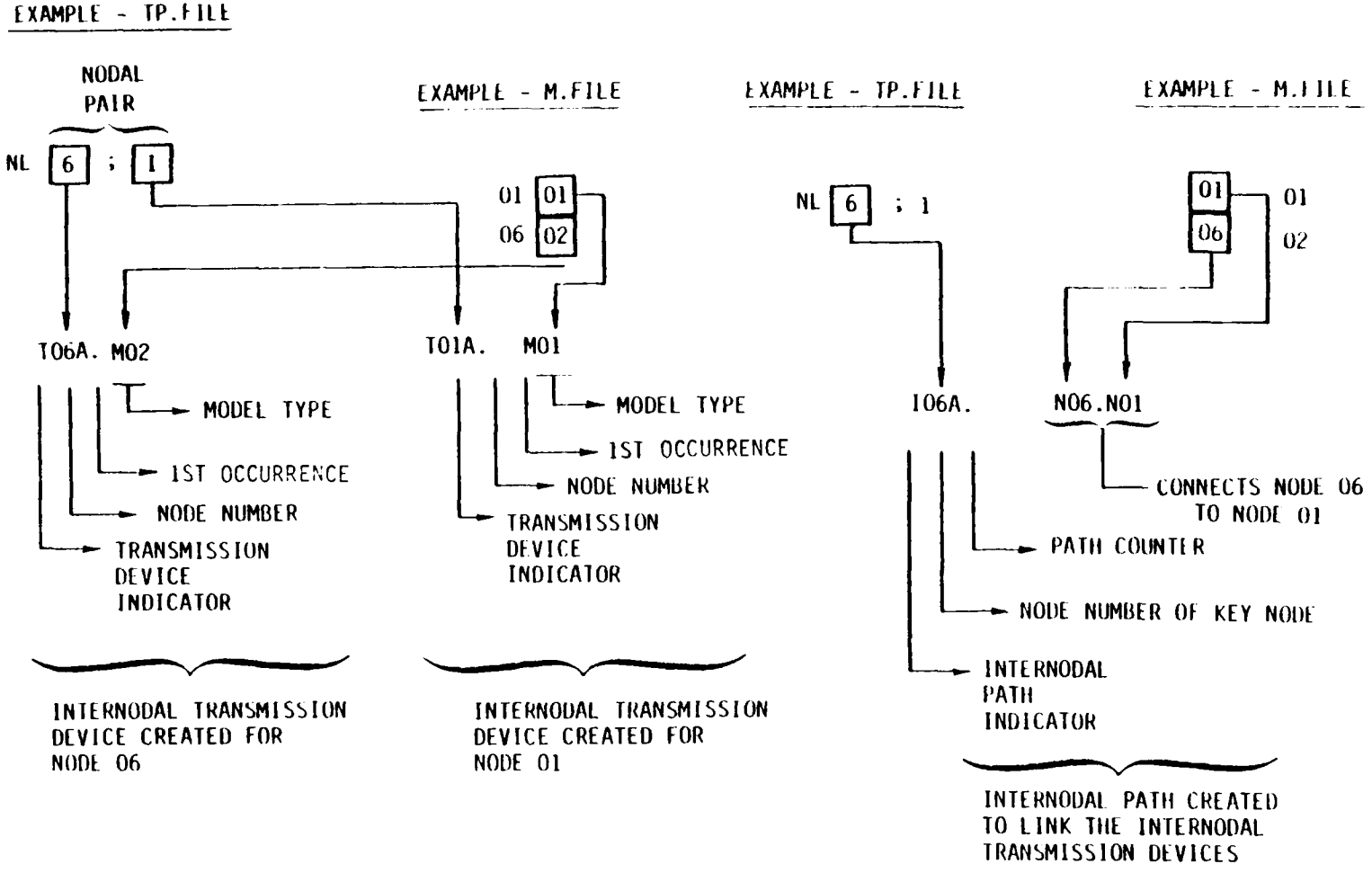
```
SPECIFY 1 T06A.M02 WHICH TRANSFERS MESSAGES AT 40000 BYTES/SEC
```

The above SPECIFY statement assumes that the default speed for the Internodal Transmission Device is 40000 bytes/sec. Should the "NL" statement connecting node 06 to node 01 contain a device speed of 34000 bytes/sec for node 01, the resulting SPECIFY statement would be:

```
SPECIFY 1 T01A.M01 WHICH TRANSFERS MESSAGES AT 34000 BYTES/SEC
```

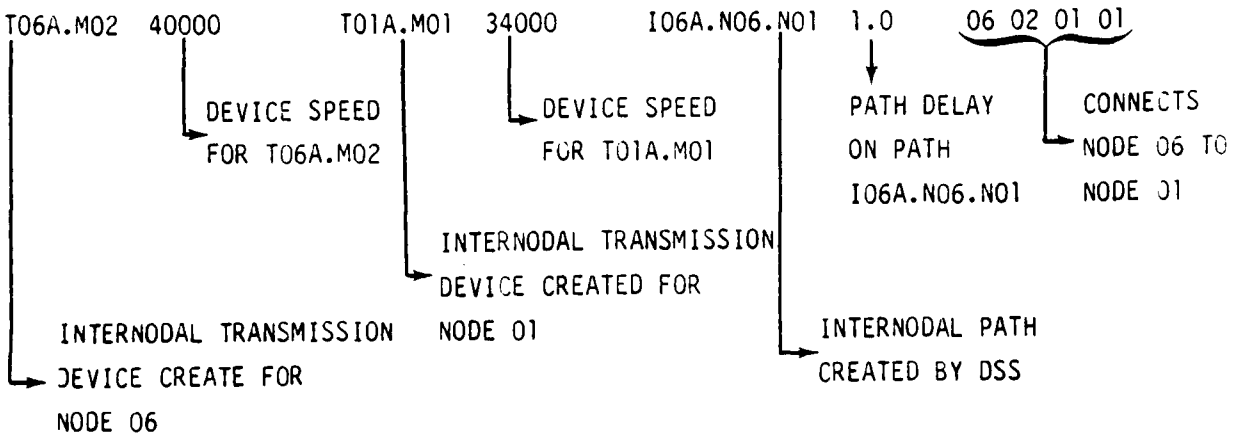
The devices and paths that DSS creates are summarized in the DSS Topology Summary Report. Each line of this report summarizes the devices and patn created for each nodal pair. Below is an example of a line from that report:

## CREATION OF AN INTERNODAL PATH BETWEEN DEVICES




3A114/27 Figure 2.4.4-1 Creation of Internodal Transmission Device

Sample line from the DSS Topology Summary Report



To summarize the DSS creation of Internodal Devices and Paths Figure 2.4.4-2 provides a capstone illustration of the DSS Topology Summary Report.

INPUTS:

TP.FILE

```
SPD 40000
NL 6 ; 1 34000 1.0 ; 7 1.5 ; 9 45000 2.0
NL 7 ; 2 40000 .75 ; 8 50000 SPD 75000
NL 8 ; 3 25000 1.0 ; 9 28000 3.0
NL 9 ; 4 18000 4.0 ; 5 2.5
```

M.FILE

```
1 01
2 01
3 01
4 01
5 01
6 02
7 02
8 02
9 02
```

M.FILE

EXEC FILE

```
MLT 0
```

EXEC file

## 2.5 DSS SIMULATION OUTPUT

As messages are transmitted from one node to the next in a DSS Simulator, statistics are collected and a report is generated when the simulation terminates. An example of a message statistics report is shown in Table 2.5-1. The header of this report is summarize below.

- Type defines a group of messages upon which message statistics are gathered. User defined.
- Total number of message associated with each message type.
- The time required for a message to traverse the network.
- Minimum delay time for each message type.
- Maximum delay time for each message type.
- Average delay time for all messages of this message type.
- Standard deviation of message delay time.

| DEVICE | DEVICE SPEED | CREATED DEVICE | SPEED | PATH CREATED PATH | DELAY | CONNECTS NODE NO. | MODEL TYPE | NODE NO. | MODEL TYPE |
|---|---|---|---|---|---|---|---|---|---|
| T06A.M02 | 40000 | T01A.M01 | 34000 | I06A.N06.N01 | .00005 | 06 | 02 | 01 | 01 |
| T06B.M02 | 40000 | T07A.M02 | 40000 | I06B.N06.N07 | .00015 | 06 | 02 | 07 | 02 |
| T06C.M02 | 40000 | T09A.M02 | 45000 | I06C.N06.N09 | .0002 | 06 | 02 | 09 | 02 |
| T07B.M02 | 40000 | T02A.M01 | 40000 | I07A.N07.N02 | .0001 | 07 | 02 | 02 | 01 |
| T07C.M02 | 40000 | T08A.M02 | 50000 | I07B.N07.N08 | | 07 | 02 | 08 | 02 |
| T08B.M02 | 40000 | T03A.M01 | 25000 | I08A.N08.N03 | .00005 | 08 | 02 | 03 | 01 |
| T08C.M02 | 40000 | T09B.M02 | 28000 | I08B.N08.N09 | .0003 | 08 | 02 | 09 | 02 |
| T09C.M02 | 18000 | T04A.M01 | 40000 | I09A.N09.N04 | .0005 | 09 | 02 | 04 | 01 |
| T09D.M02 | 18000 | T05A.M01 | 40000 | I09B.N09.N05 | .00035 | 09 | 02 | 05 | 01 |

Figure 2.4.4-2 DSS Topology Summary Report

In addition, DSS collects these same statistics on a system wide basis for all messages.

Message characteristics, such as message length, are defined in the MESS file. For each type of message defined there the user has the option of specifying what statistics gathering group (SGG) it belongs to. The number following the "STT" keyword defines the group. For example, three typical lines in the MESS file might be:

```
1   2    MML 30.          STT 1 ;
1   3    MML 40.          STT 2 ;
1   4    MML 50.          STT 1 ;
```

In this example the messages defined in lines 1 and 3 belong to the same Statistical Gathering Group. These messages will be treated as one group for statistics gathering purposes. The second line defines a second SGG. In this way, by defining an arbitrary number of SGG's, the user can decide how detailed the report generator should be in collecting separate statistics on sub-classes of messages in the network. The user must specify the maximum number of SGG's used in a particular run by using the NSG Keyword in the EXEC file. The default is that only system wide statistics will be generated.

Table 2.5-1
Message Statistics Report

MESSAGE STATISTICS

| TYPE | NO. OF MESSAGES | DELAY TIME MIN | MAX | AVE | STD |
|---|---|---|---|---|---|
| 1 | 27 | 3.70 | 31.26 | 17.43 | 8.50 |
| 2 | 39 | 2.70 | 32.80 | 14.64 | 7.52 |
| 3 | 26 | 1.70 | 30.10 | 15.75 | 8.58 |
| 4 | 8 | 1.40 | 6.40 | 4.00 | 1.60 |
| 5 | 33 | 1.71 | 24.11 | 13.26 | 6.76 |

SYSTEM - WIDE

| NO. OF MESSAGES | DELAY TIME MIN | MAX | AVE | STD |
|---|---|---|---|---|
| 133 | 1.40 | 32.80 | 14.44 | 8.13 |

Since DSS programs are translated into ECSS and SIMSCRIPT, all of the ECSS statistics gatnering routines and report generators are available to DSS. Since each node in the network simulator has its own unique device, path, job and process names, the statistics collected from the simulator are quite detailed. A summary of these reports is given below. (Refer to [FEDE 76] for a complete description).

ALLOCATION REPORT

This report provides utilization statistics as well as average and maximum allocation times for all private devices at every node in the simulator.

ALLOCATION QUEUE REPORT

This report provides all of the queueing statistics associated with private devices including:

- Total number of requests
- Average wait for allocation
- Total enqueued
- Average queue length
- Maximum time in queue
- Percent time empty

EXECUTION REPORT

The Execution Report gives the percent of time that a processor device was busy by specific jobs that executed on that device. This is an excellent way to determine what jobs are requiring the most processor time.

PROCESSOR UTILIZATION REPORT

This report gives the utilization of all processors in the network as well as the number of activations for each device during the reporting period.

EXECUTION (READY) QUEUE REPORT

The Execution Queue Report displays statistics on number of arrivals to a processor and all of the queueing statistics including average and maximum wait for service, average and maximum queue length and percent time empty.

JOB STORE REPORT

The Job Store Report shows statistics for each storage device on:

- Number of jobs loaded
- Average number of jobs loaded
- Job loaded times
- Batch capacity

BATCH JOB STORE QUEUE REPORT

For all storage devices in the network this report provides:

- Total number of requests for a given device
- Average wait for loading
- Average and maximum queue lengths
- Percent time empty

TRANSMISSION PATH REPORT

The Transmission Path Report summarizes how busy a particular path was for the reporting interval. It includes the total number of transmissions across the path, and the average and maximum number of messages transmitted across the path at any one time. Also, average transmission lengths and percentage utilization are reported.

TRANSMISSION PATH QUEUE REPORT

This report provides all of the queueing statistics associated with any path in the network. It includes:

- Total number of requests during the report interval
- Average wait for transmission
- Total enqueued
- Average queue length
- Maximum queue length
- Average and maximum time in queue
- Percent time empty

TRANSMISSION REPORT

This report displays statistics on the amount of traffic a transmission device handled. Included are total number of transmissions, average number of transmissions, transmission capacity and the average and maximum transmission lengths. The percentage of the transmission capacity used during the reporting interval is also provided.

CRITICAL DEVICE TRANSMISSION QUEUE REPORT

This report is an excellent means of locating bottlenecks in the transmission medium of networks. A critical device on a path is the device which causes a message to be enqueued before it can be transmitted. Any device which is a critical device by this criterion at any time during the simulation is included by name in this report. Specifically, this report includes:

- Device name
- Total enqueued
- Average queue length
- Maximum queue length
- Average and maximum time in queue
- Percent time empty

TRANSMISSION RATE REPORT

The transmission rate report gives the total number of messages which were transmitted on a particular device. In addition it gives the average cumulative transmission rate for a device during the reporting intervals. The cumulative rate is the sum of the rates of the transmissions occurring simultaneously on the device. The average and maximum transmission rate per message is provided with the percent idle time.

STORAGE REPORT

For storage devices this report provides these statistics:

- Total number of requests
- Average and maximum storage utilization
- Average and maximum storage request size
- Storage capacity
- Percentage utilization

STORAGE QUEUE REPORT

All of the queueing statistics associated with storage devices are summarized in this report. These include total requests, average wait for service, average and maximum queue lengths, average and maximum queue times and percent time empty.

PROCESS/JOB REPORT

There is a very extensive report put out on each of the external processes and jobs in the Workload Description section for each of the nodes in the network simulator. This report includes, by process or job name:

- Total number of instances of the process or job
- Number of completed instances
- Average instance length
- Maximum instance length
- Total length of instances
- Average execution time per instance (Jobs only)
- Maximum execution time per instance (Jobs only)
- Total execution time per instance (Jobs only)
- Average transmission time
- Maximum transmission time
- Total transmission time per instance
- Average time blocked for loading (Jobs only)
- Maximum time blocked for loading (Jobs only)
- Average time blocked for activation (Jobs only)
- Maximum time blocked for activation (Jobs only)
- Average time blocked for transmission
- Maximum time blocked for transmission
- Average time blocked for allocation
- Maximum time blocked for allocation
- Average time blocked for storage
- Maximum time blocked for storage

# SECTION 3
# DEVELOPING MODELS OF COMPUTER NETWORKS

## 3.1 GENERAL METHODOLOGY

The purpose of this section is to provide an overview and introduction to the technical approach used in the development of the high level and detailed level computer network simulations.

Before all major design concerns can be identified, one of the most important initial considerations must be what types of analyses the models are going to support. It is necessary to understand the intended use of a simulation model to effectively design it. The way a model is to be used or the types of analyses it will support affect the level of detail of the model, what aspects of a system will be represented, the format and content of the output reports, and the inputs required. In other words, the purpose of a simulator is to answer certain performance and/or cost-related questions. This is the driving factor in simulator design.

Once the intended use has been defined, a first approximation can be made as to the level of detail that should be incorporated into the model. From past experience with models of computer systems and networks, a continuum of detail levels can be described. Any particular model is just one point on that continuum. What is important is to provide a simulation tool that has the flexibility to easily and quickly build models along a wide portion of that spectrum. This is necessary because it is not always possible to determine a priori the appropriate level of detail to answer certain questions. For example, a system-level model may contain a disk subsystem model as one component (see Figure 3.1-1). Under lightly loaded conditions file access time may be adequately represented as a simple delay. However, if the file transfer times in this model are short compared with the known overhead involved, such as disk latency times, then it may be necessary to explicitly account for this overhead. That is, it may not be feasible to assume one overall transmission delay in order to estimate job response times and device utilizations with an acceptable degree of accuracy. What is clearly needed is

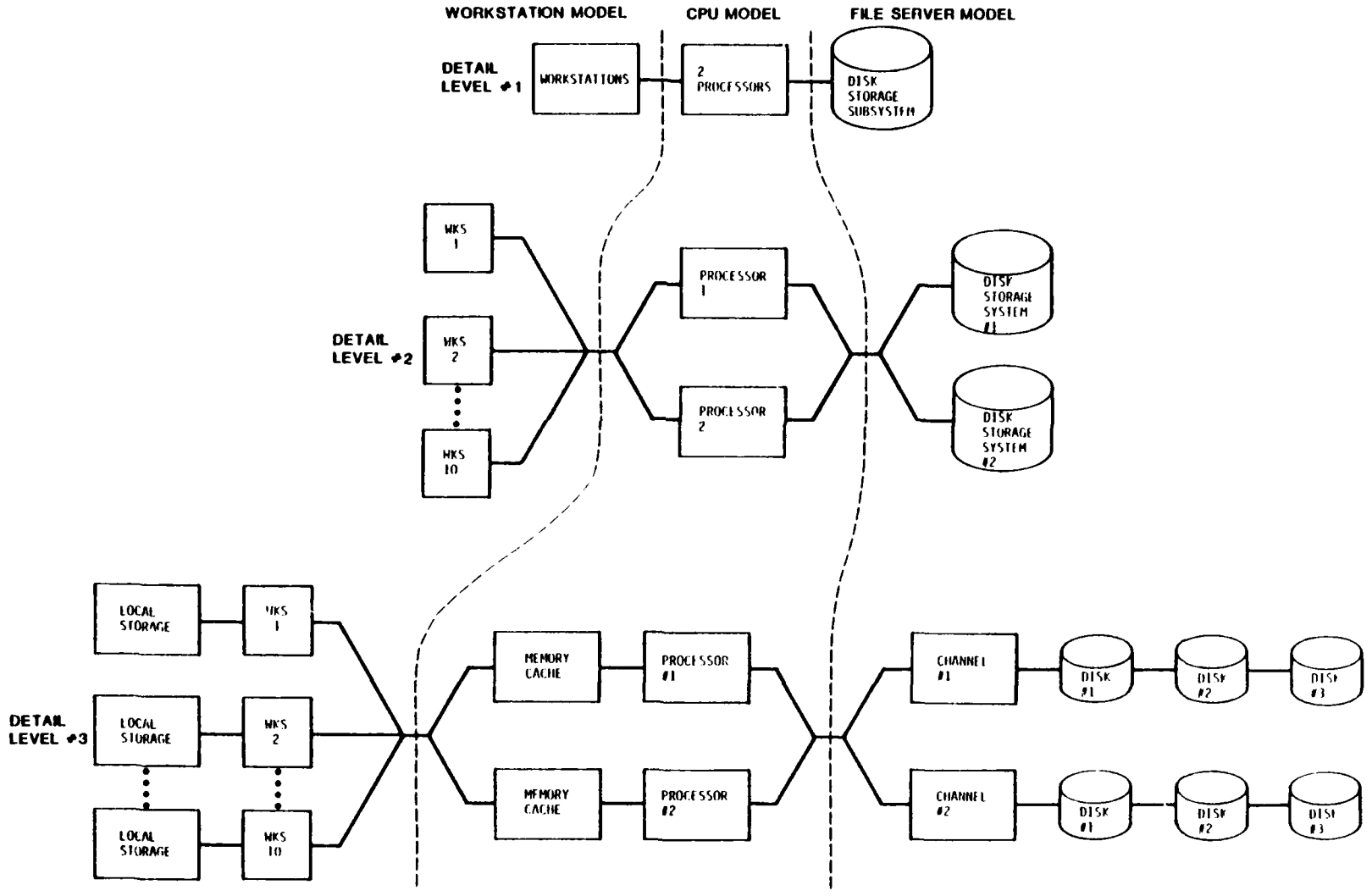



Figure 3.1-1 Three views of a Simulated Computer System

a simulation tool that provides the user with an effective means of easily incorporating different levels of detail within a model to help solve two related classes of problems:

- Sensitivity analysis
  To determine those critical system components (both hardware and software) that, when varied, have a significant impact on the performance of the system.
- Tradeoff analysis
  To estimate the accuracy of performance measures (e.g., throughput, response times, etc.), given certain run time constraints (and therefore detail level constraints) of the simulator.

For these reasons, it is unsatisfactory to have a simulation tool that can model systems along a narrow band in the spectrum of possible detail levels. In addition, computer and software vendors are providing new products on a regular basis. Here again it is not possible to predetermine the level of model detail that will be required to analyze these new offerings.

## 3.2 MODULARITY AND RECONFIGURABLE SIMULATORS

As described in Section 2.3.1 DSS allows the user to build models of nodes in a system and save each model separately in a model library.

By having direct correspondence between models in the simulator and nodes in the simulated system, models can be mixed and matched, depending on the type of experiments that are to be performed. For instance, Figure 3.1-1 depicts three highly simplified views of a simulated computer system. Going from top to bottom in this figure, more detail is included in the model. There is a total of nine models stored in the model library: a workstation model, a CPU model, and a file server model for each of the three detail levels. Figure 3.1-1 shows only the increased complexity of the hardware configuration; there could also be a corresponding increase in the detail of the resource managers (operating system models) and in the workload characterizations. To test the sensitivity of the file access mechanisms on the job throughput rate, the workstations would act only as sources and sinks for generated message flows.

Then, to answer questions about the file access mechanism, the workstation model and processor model of detail level one and the file server model of detail level three could be used, while varying the file access methods. To answer other types of questions, the simulator can be reconfigured in a similar manner.

In the more traditional approach, a single simulator is designed and built with the object of answering all of the questions that motivated the simulation study at the outset. Building DSS simulators, we can be more selective in our approach. The simulator can be reconfigured depending on the requirements of a particular subset of the proposed experiments. This makes the experimentation stage less time-consuming in terms of run time costs, and the simulation output is reduced and directly applicable to the questions the experiments were designed to answer. DSS has been used with this approach in building simulators of computer networks. The key to this approach is that DSS supports the building of simulators in a highly modular fashion. As can be recalled from the example in Section 2.2.5, every nodal model is constructed from three basic components that may be built separately: the System Description, Workload Description, and Resource Manager sections. Each nodal model is built separately, but may be connected to other nodal models at a later time to form one large network-wide model. This structure is depicted in Figure 3.2-1. This modular construction enables the user to move easily from one level of detail to another. For example, it would take only a few minutes to describe, using the model language of Section 2.3, the configurations depicted in Figure 3.2-1.

Similarly, the Workload and Resource Manager sections can be created or used from a model library. In this way, DSS provides reconfigurable simulators based on the modularity of its design approach.

## 3.3 VERIFICATION, VALIDATION, AND CALIBRATION

When developing and testing models of computer systems and networks, verification, validation and calibration are three essential elements in determining the usefulness of the models. These processes are usually done on an iterative basis. For example, in modeling the job arrival process to a

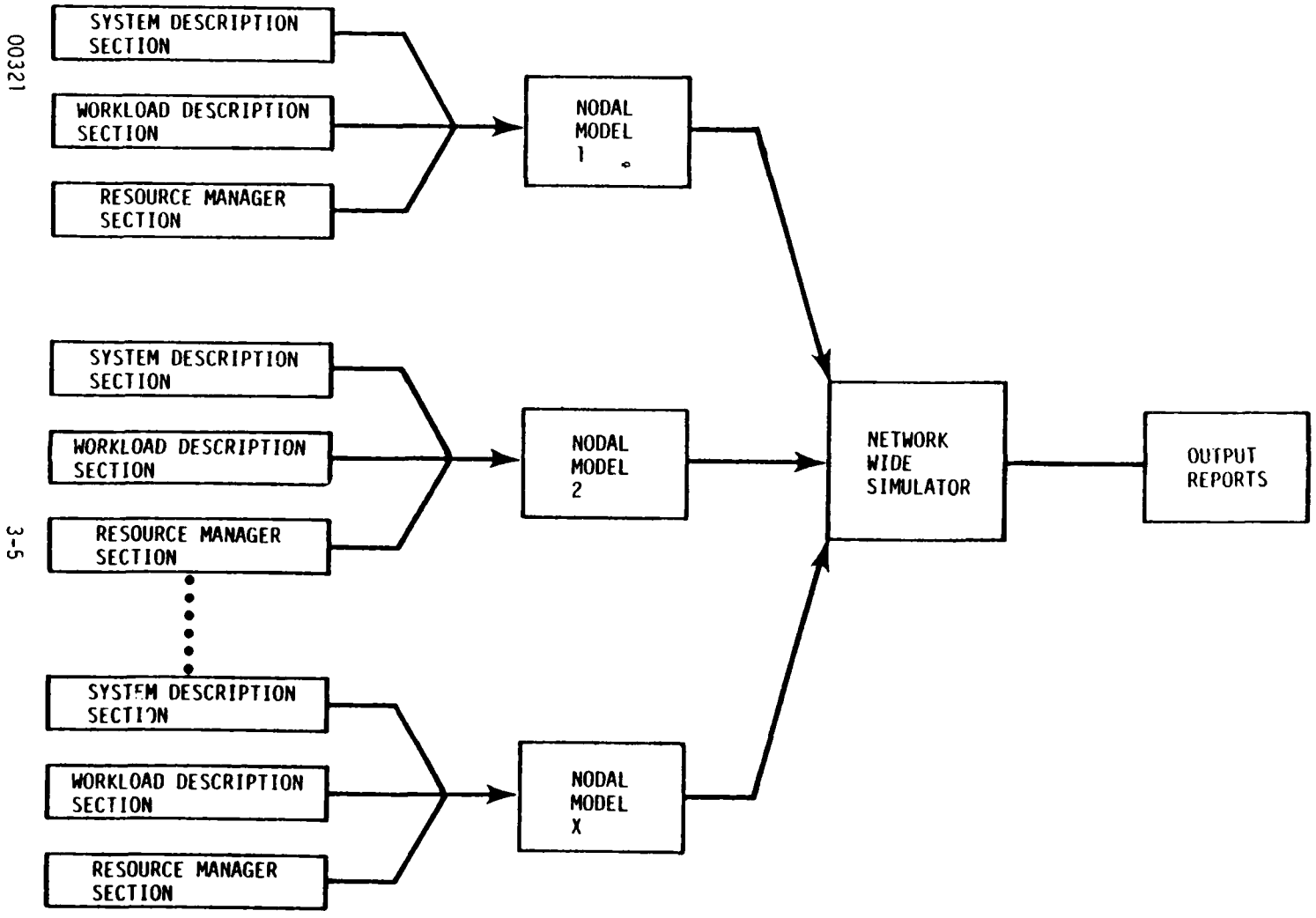



Figure 3.2-1 Modular Construction of Network Simulators

computer system by a Poisson process, one is first interested in whether or not the simulator is generating jobs with an interarrival time that comes from an exponential distribution (verification). Once this has been established, the rate of the Poisson process may be adjusted to agree as closely as possible with the rate suggested by the empirical data (collected, perhaps, by a software monitor); this is the calibration stage. Finally, validating the model might consist of checking the output from the simulator against a range of data collected from several samples. If the agreement is not satisfactory, a new distribution may be suggested by the data, and the verification process would begin again. Usually with developed simulation tools it is not necessary to test the random number generators although this can be a useful exercise.

The design of DSS has taken into account the verification, validation and calibration stages of simulators by providing special capabilities for the analyst in these stages of model development. In the following three sub-sections we describe in greater detail what is entailed in verification, validation and calibration and the way in which DSS can be used in all three stages.

### 3.3.1 VERIFICATION

In this initial stage, one need not be concerned with how well the model represents the real system; the principal interest is in the ability of the simulator to represent the model structure. The verification process used for the simulators in the system-level and data-transfer-level models can be outlined in four steps.

- Internal Consistency

  This step involves checking to ensure that all of the model elements have the minimum number of required parameters; that allocated resources are later deallocated and, in general, that the flow of information in the system is not impeded by the simulator structure.

- Exogenous Variables

  This step in the verification process ensures that the simulator is driven by processes and loading parameters that are consistent with the model design. For instance, a workload description that does simulated disk reads and writes is not appropriate for a high-level model whose only elements are switching nodes and channels.

- Trace reports

  There are two classes of trace reports that are of primary importance. Simulator programs may be viewed as a hierarchically structured set of interacting processes. The first trace facility allows for checking that the intended hierarchical structure is indeed being implemented: only certain processes may initiate other processes. The second trace facility produces a detailed listing of elementary operations and the simulated time at which each one occurred. This microscopic view of the simulator can be turned on and off when desired and is an invaluable aid in debugging and verifying simulators.

- Dump Reports

  Simulator programs have many internally defined sets and tables that are usually transparent to the user. For instance, there is a global set called the staging list for jobs that are waiting for execution time on simulated processing devices. The state of these sets and tables may be interrogated at random times and dumped to a specified file for later analysis. A dump report is of help in verifying that the simulator is performing as planned, and not just producing acceptable results because of a peculiarity in the loading parameters or system description.

3.3.2 AN AID TO VERIFICATION: THE EVOLUTIONARY APPROACH TO MODEL DEVELOPMENT

As already stated, the DSS simulators model each node of a network separately, and the nodal models are combined to simulate the entire network. Network simulators can become quite complex and difficult to debug as new nodes are added. The ideal situation would be to have a network simulator grow incrementally as new nodes are added, with each new addition debugged separately. This approach has been used with the high level and detailed level simulators.

Three high-level models of networks were built using the facilities of DSS. Table 3.3.2-1 summarizes the characteristics of each of these models. These models are of packet-switched (store-and-forward) networks. Going from Model 1 to Model 3 (see Table 3.3.2-1) the architecture of a node, routing algorithms, flow control procedures, and data formats become more detailed.

The purpose of this section is not to go into the particulars of each one of the models, but to describe how an evolutionary approach to model development can greatly reduce the time needed to build and verify simulators.

Three levels of network complexity were identified for each one of the high level models. In level A (Figure 3.3.2-1), the simplest network is described: a two-node network in which one host communicates directly with another host site. In this configuration, only one model from the DSS model library is used, since the nodes are identical. Trace output, one of the most effective means of verifying the behavior of a simulator, can be voluminous for a multinode network simulator. By starting with very simple configurations, it is possible to take very detailed looks at the behavior of the simulator while keeping the trace output to a manageable level. In complexity level B (Figure 3.3.2-1), a switching node site is added. This three-node configuration requires two models from the DSS model library, one for the switching node and one for the host sites. Again, keeping things as simple as possible increases the chances of detecting problems at an early stage. The third level of complexity, level C (Figure 3.3.2-1), is an arbitrary network configuration using the verified models from complexity levels A and B. The changes to the TP.FILE and M.FILE are quickly made when going through these different levels. For all three high-level models, this approach to model development was instrumental in producing completely debugged simulators by the time complexity level C is reached.

### 3.3.3 CALIBRATION/VALIDATION

After a simulator has been verified, a set of validation experiments is

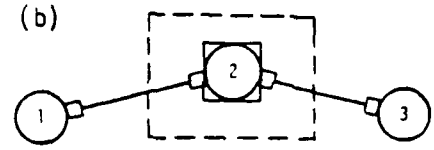
carefully designed to determine whether or not the model, which is implemented by the simulator, conforms to expectations. If the agreement is not good, the parameters or the structure of the model is manipulated and the resulting output data are again compared with independently generated historical data. This procedure, calibration of a model, is continued until the model and historical data agree within a predetermined tolerance. This procedure, however, does not necessarily produce a valid model since it may only be representative of a particular set of input data. A successfully calibrated model must then be compared with other sets of historical data to ensure that the model is sufficiently general to be a predictor of behavior across a broad range of input data.

Table 3.3.2-1 Summary of Key Model Characteristics

| CHARACTERISTICS | HIGH-LEVEL MODEL 1 | HIGH-LEVEL MODEL 2 | HIGH-LEVEL MODEL 3 |
|---|---|---|---|
| MODEL ARCHITECTURE | • PROCESSOR<br>• INTERNODAL TRANSMISSION DEVICES | • PROCESSOR<br>• CHANNELS<br>• TERMINALS<br>• BUFFERS<br>• INTERNODAL TRANSMISSION DEVICES | • PROCESSOR<br>• CHANNELS<br>• TERMINALS<br>• BUFFERS<br>• INTERNODAL TRANSMISSION DEVICES |
| ROUTING | FIXED | FIXED | NONDETERMINISTIC ADAPTIVE |
| DATA FORMAT | MESSAGE | PACKET | PACKET |
| FLOW CONTROL | NO | YES | YES |
| MESSAGE INTERARRIVAL TIME | CONSTANT | CONSTANT OR STOCHASTIC PROCESS<br>- UNIFORM DIST.<br>- EXPONENTIAL DIST. | CONSTANT OR STOCHASTIC PROCESS<br>- UNIFORM DIST.<br>- EXPONENTIAL DIST. |
| MESSAGE LENGTH | CONSTANT | CONSTANT OR STOCHASTIC PROCESS<br>- UNIFORM DIST.<br>- EXPONENTIAL DIST. | CONSTANT OR STOCHASTIC PROCESS<br>- UNIFORM DIST.<br>- EXPONENTIAL DIST. |


TP.FILE
SPD 35000
NL 1; 2

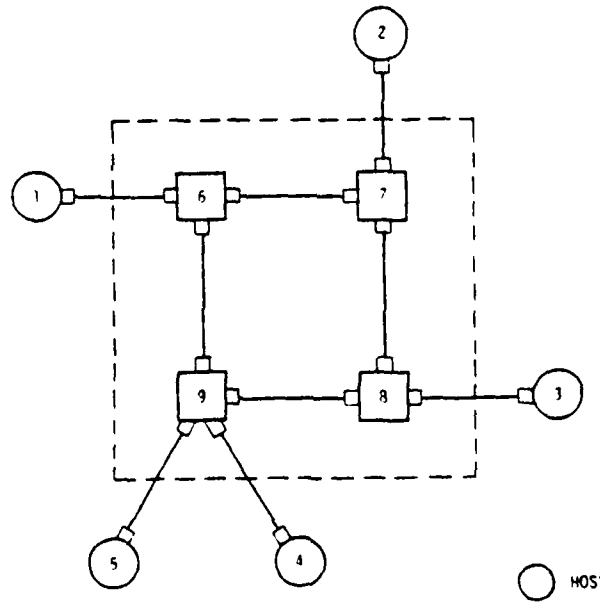
M.FILE
1 01
2 01




TP.FILE
SPD 35000
NL 1; 2
NL 2; 3

M.FILE
1 01
2 02
3 01

24131/6




TP.FILE
SPD 35000
NL 6; 1; 7; 9
NL 7; 2; 8
NL 8; 3; 9
NL 9; 4; 5

M.FILE
1 01
2 01
3 01
4 01
5 01
6 02
7 02
8 02
9 02

2A131/5

Figure 3.3.2-1 Complexity Levels
A. Complexity Level A - Host to Host Configuration
B. Complexity Level B - Host/Switch/Host Configuration
C. Complexity Level C - Arbitrary Configuration

Below is an outline of some of the steps in the validation and calibration process for computer network simulations.

- Simplified Workload Description

  In this step, the loading factors are minimized to include only one or a very few jobs. By simplifying the workload description, it is possible in many instances to determine quite accurately what the expected values of the performance measures should be. The complication of resource contention is practically nullified, and the job flow can usually be more easily determined. Characteristics of the models not load dependent can be validated in this step.

- Deterministic Models

  Some models may be more easily analyzed and performance measures calculated if the random variates that determine key parameters of the system, such as interarrival times, are assumed to be deterministic. This step may be used with the simplified workload description to validate some of the performance measures of the simulator. For example, routing protocols may be validated in this way, since they do not depend on the distribution of job interarrival times, but on the congestion and availability of switching nodes.

- Analytic Models

  Analytic models can provide an independent source of performance measures. For instance, a communication network in the high-level model may be simulated as a set of nodes (switching computers) and connected by high-speed channels. Assuming the service rates of the nodes and channels are exponentially distributed with a Poisson stream of jobs as input to designated nodes, this system can be modeled as a Markovian network of queues and servers. This allows derivation of performance measures, such as utilization and message response time, independently of the simulator. These measures may then be used to validate the simulator. Analytic models, when they are mathematically tractable, will be used in the proposed validation experiments.

- Hypothesis Testing

  Certain systems have known or expected behavior characteristics that can be tested using the simulator. This expected behavior may be known in only a qualitative sense. For instance, it might be determined that utilization of specified resources cannot be improved in a certain system due to capacity constraints. Another quantitative hypothesis might be that, as throughput increases, response time is degraded. These hypotheses, even though they cannot be stated precisely in quantitative terms, can be tested using the simulator. If there is disagreement between the hypotheses and the simulator output, either the fault in the simulator must be corrected or an explanation of why a hypothesis is false must be found. Usually in this stage other subsidiary hypotheses concerning the behavior of the system are suggested, which can be checked in turn by the simulator.

- Independent Random Number Generation

  In analyzing the output from a simulator during the calibration/validation phases we are interested in the true performance measures of the modelled system independently of the noise generated by using a particular random number stream. Simscript II.5 provides ten starting seeds for this purpose. This allows for greater flexibility in the design of experiments and reduces the cost of simulation runs without reducing the usefulness of the results. In representing random processes in a model, it is important to be able to isolate various processes. For example, in a simple queuing system, one expects that the arrival distribution should be independent of the service mechanism. If we are limited to one sequence of random numbers, it would be difficult to represent this independence. (It could be done, to a degree, with very long runs.) However, by assigning different random number streams to the two processes and thereby starting from different values, we can have the desired independence. Further, from an experimental design point of view, we can more quickly demonstrate the effects of a change in value of a controlled variable. For example, if we change the number

of servers in our simple queueing system and repeat the experiment with the same starting seeds for each stream, then, through careful implementation of the model, we can show that any changes in the results from the simulations are the direct result of the changes in the controlled variable (number of servers) and not due to random variations. [RUSS 82].

Since there is literally an unlimited number of models that can be implemented using the simulators for the high level and detailed level models, only a subset of these possibilities can be tested. However, the number of different types of elements in these simulators is finite and well defined. For instance, there are computing resources and input devices in the high level model. All of the elements have been tested by applying different loading factors over a broad spectrum of values to ensure that the utilization of the system resources goes from lightly loaded to nearly 100 percent of their capacity. The library routines that implement standard rescurce managers, communication protocols, etc. have also been tested in this way.

The models that have been developed are documented in the order of increasing complexity, and the loading factors applied to them are described to ensure that all combinations of model elements have been verified.

# SECTION 4
# HIGH LEVEL MODELS OF COMPUTER NETWORKS

## 4.1 *INTRODUCTION*

In Section 4 each of the High Level Models (HLM) is described - the input files that are required, the architecture and functional flow of each model and the output generated by each model.

HLM #1 simulates a message switched network. Messages are generated at host or user sites, routed through the network by a deterministic routing algorithm and finally destroyed at the destination site. HLM #2 has all of the facilities of HLM #1. In addition, it incorporates two types of flow control procedures to control congestion within the network. The model architecture explicitly represents input/output devices at the host sites and buffer sizes at every node in the network. Also, messages may be broken into packets and routed through the network in this format. HLM #3 is similar to HLM #2. However, HLM #3 simulates adaptive routing procedures through a non-deterministic routing table. See Table 4.1-1 for a summary of these differences. Statistics representing message delays, resource utilization, bottlenecks in the system, queueing etc. are collected and reports generated for each node in the network.

As we move from HLM #1 to HLM #3 the complexity of the models increases, which requires more detailed input data from the user. The models are upwardly compatible. For example the routing table for HLM #1 may be used as an input file to HLM #3. By providing a range of models of increasing complexity the network designer is not constrained to inputting model parameters at only one level of detail. For instance, buffer size is a required input for HLM #2. If this data is not available the designer may use HLM #1 where this does not play a factor in the model inputs.

Table 4.1-1 Summary of Key Model Characteristics

| CHARACTERISTICS | HIGH LEVEL MODEL 1 | HIGH LEVEL MODEL 2 | HIGH LEVEL MODEL 3 |
|---|---|---|---|
| MODEL ARCHITECTURE | • PROCESSOR<br>• INTERNODAL TRANSMISSION DEVICES | • PROCESSOR<br>• CHANNELS<br>• TERMINALS<br>• BUFFERS<br>• INTERNODAL TRANSMISSION DEVICES | • PROCESSOR<br>• CHANNELS<br>• TERMINALS<br>• BUFFERS<br>• INTERNODAL TRANSMISSION DEVICES |
| ROUTING | FIXED | FIXED | NON-DETERMINIST ADAPTIVE |
| DATA FORMAT | MESSAGE | PACKET | PACKET |
| FLOW CONTROL | NO | YES | YES |
| MESSAGE INTERARRIVAL TIME (MIT) | CONSTANT | CONSTANT OR STOCHASTIC PROCESS<br>- UNIFORM DIST.<br>- EXPONENTIAL DIST. | CONSTANT OR STOCHASTIC PROCESS<br>- UNIFORM DIST.<br>- EXPONENTIAL DIST. |
| MESSAGE LENGTH (MML) | CONSTANT | CONSTANT OR STOCHASTIC PROCESS<br>- UNIFORM DIST.<br>- EXPONENTIAL DIST. | CONSTANT OR STOCHASTIC PROCESS<br>- UNIFORM DIST.<br>- EXPONENTIAL DIST. |

## 4.2 MODEL SPECIFIC INPUT FILES

Several model specific input files are required to run ECSS simulations. They are:

- The EXEC file which provides the defaults required for the simulation,
- The ROUT file which provides the routing table for the simulation and
- The MESS file which defines all message characteristics.

The format of these input files are designed to be upwardly compatible so as to require little or no alternations when running any of the high level models. For example, the ROUT file used for High Level Model 1 can be used without any changes when running High Level Model 3.

The dashed box in Figure 4.2-1 identifies the Model Specific Input files. These files are detailed in the DSS User's Manual.

## 4.3 HIGH LEVEL MODEL 1

High Level Model 1 simulates a message switching network. Messages are routed through the network by a deterministic, fixed routing algorithm. This model contains no flow control procedures in that when a message is transmitted it will always be accepted by the receiving node.

### 4.3.1 MODEL ARCHITECTURE

High Level Model 1 requires two DSS model types:

- A model to describe the behavior of the host sites and
- A model to describe the behavior of the switching nodes.

The association between the model type and node number is defined in the M.FILE.

#### 4.3.1.1 Host Sites

The function of the host site is to generate new messages and destroy messages sent from other host nodes. Each host site consists of:

- A processor;

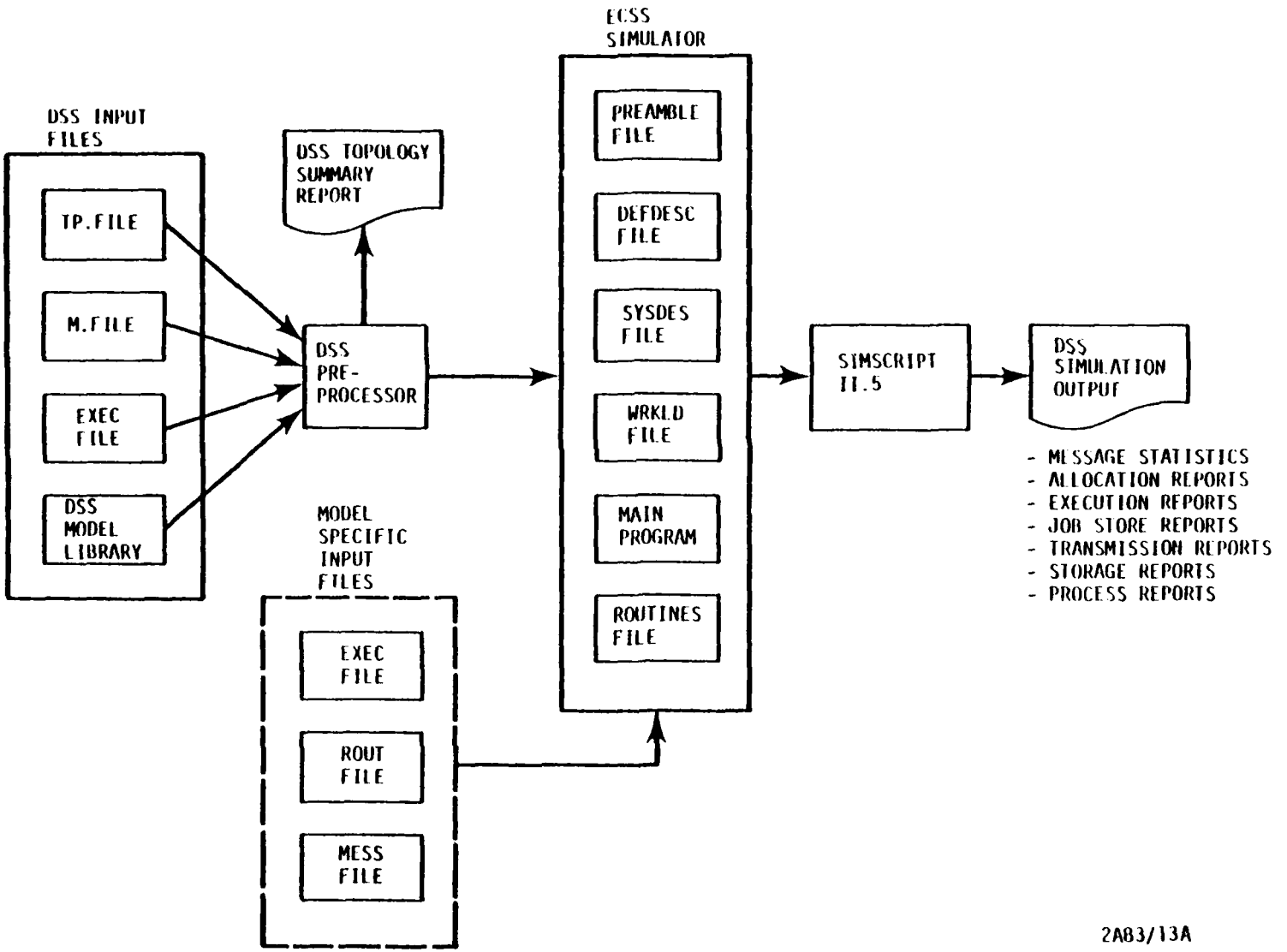



Figure 4.2-1 Model Specific Input Files

- A variable number of internodal transmission devices. The number of devices will vary depending on whether the option is multiplexed or dedicated channels. The example shown in Figure 4.3.1.1-1 is dedicated. Therefore, there will be as many internodal transmission devices as there are paths entering or leaving each node.

Figure 4.3.1.1-1 shows the host site configuration.

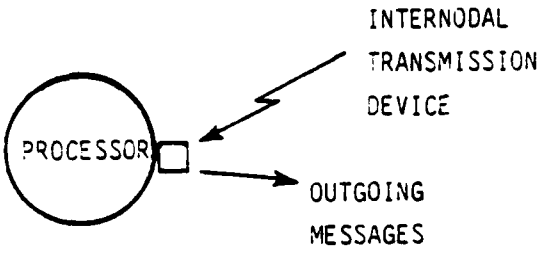



Figure 4.3.1.1-1 Host Site Configuration

The specifications describing the Processor, located at each host site, are defined in the DSS Model Library. To change these specifications the user must change the DSS Model.

The Internodal Transmission Device specifications are parameterized directly through the TP.FILE. Table 4.3.1.1-1 summarizes the System Description for a typical nost site.

4.3.1.2 Switching Nodes

The function of the switching node is to accept and forward messages along the network. Each switching node consists of:

- A processor
- A variable number of Internodal Transmission Devices. Refer to Host description

Figure 4.3.1.2-1 is a representation of a switching node.

Table 4.3.1.1-1 System Description for Host Site

| DEVICE NAME | ECSS DEVICE TYPE | PERFORMANCE SPECIFICATIONS |
|---|---|---|
| D01A.CPU (Processor) | EXECUTION AND JOB STORE DEVICE | 500000 INSTRUCTIONS/SEC |
| T01A.M01 (Internodal Transmission Device) | TRANSMISSION DEVICE | 15000 BYTES/SEC |

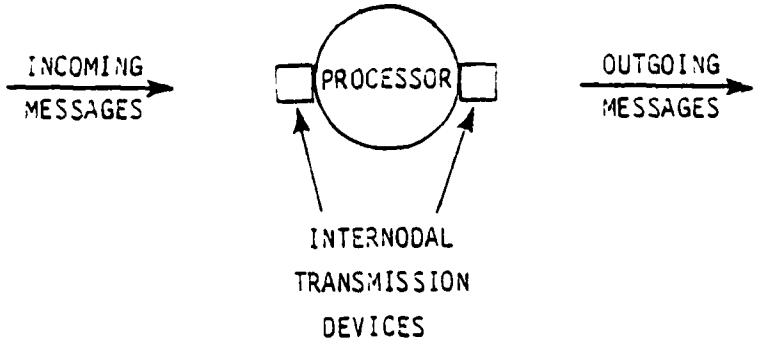



Figure 4.3.1.2-1 Switching Node Configuration

Specifications describing the processor are again defined in the DSS Model Library. The specifications related to the Internodal Transmission devices are defined in the TP.FILE. Table 4.3.1.2-1 summarizes the system description for a typical switching node. The table also shows the internodal path connecting two internodal transmission devices from node 1 and node 2.

Table 4.3.1.2-1 System Description for Switching Node

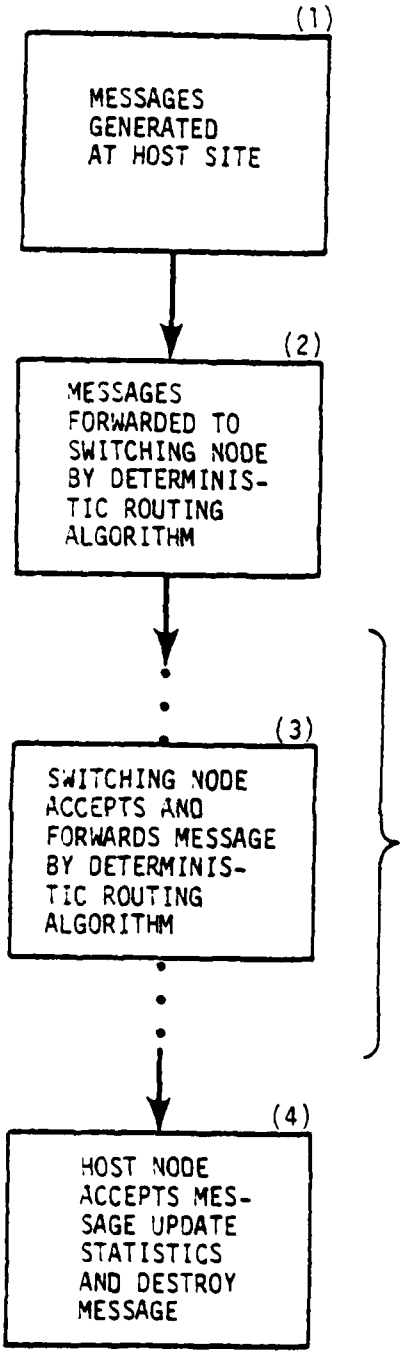
| DEVICE NAME | ECSS DEVICE TYPE | PERFORMANCE SPECIFICATIONS |
|---|---|---|
| D02A.CPU (Processor) | EXECUTION AND JOB STORE DEVICE | 400000 INSTRUCTIONS/SEC |
| T02A.M02 (Internodal Transmission Device) | TRANSMISSION DEVICE | 10000 BYTES/SEC |
| P01A.N01.N02 (Internodal) | PATH | CONNECTS T01A.M01 to T02A.M02 |

4.3.2 FUNCTIONAL LOGIC FLOW

High Level Model 1 simulates a forward switching network. Host sites generate and forward messages to a switching node. Switching nodes accept and forward messages to either another switching node or to a host site. Passing messages between switching nodes may require several transfers depending on the topology of the network. Finally, the final destination host site accepts the message, updates the message statistics and destroys the message. Figure 4.3.2-1 is a functional flow diagram of this process. Two major external processes provide the control functions (eg., message generation and starting computer jobs) for High Level Model 1. They are:

- ..ARR process (message generation) and
- CP Process (communication process - controlling computer jobs)

A brief description of each process is provided:




- ROUTINE RD.MSG.AR AND EXTERNAL PROCESS .ARR COMBINE TO ACCOMPLISH THIS TASK
- ECSS JOB JU PROVIDES THIS FUNCTION
- MAY REQUIRE SEVERAL MESSAGE TRANSFER DEPENDING ON NETWORK TOPOLOGY.
- ECSS JOB JW ACCOMPLISHES THIS TASK
- ECSS JOB JT PROVIDES THIS FUNCTION

Figure 4.3.2-1 Functional Logic Flow of High Level Model 1

2A83/7

Message Generation - External Process ..ARR

The ..ARR process provides the vehicle for generating every message introduced into the network. This process simply cycles according to the interarrival rate (MIT) established in the MESS file or defaulted in the EXEC file. Messages are created each time the process cycles. The number of instances of ..ARR in the External Process Reports indicates the number of different message types generated during the simulation run.

Communication Process - External Process CP

The primary function of the communication process is to control the communication between nodes. A communication process may start jobs on a processing unit which simulates the activity of transferring, switching and receiving messages. Statistics on these external processes are displayed on the External Process Report. The number of instances for the CP processes should be one. These processes are started only once and continue to operate for the full duration of the simulation.

The communication process activates and controls three computer jobs:

- JOB JU
- JOB JW
- JOB JT

A description of these jobs is provided.

JOB JU - Message Transferring

JOB JU simulates the activity of transferring a message from a host site to a switching node. It accomplishes this task by filing the pointer to the message (MSG) data in the switching nodes message file (..MGFILE). The JOB concludes by signaling the switching node. This notifies the communication process that it has a message waiting for it. The total number of instances for JOB JU can be interpreted as the total number of messages transmitted from JU's host site.

JOB JW - Message Switching

Message switching is accomplished by activating JOB JW. This job is identical in function and logic to job JU except it is initiated in the communication process of a switching node. The total number of instances for JOB JW can be interpreted as the total number of messages switched at JW's switching node.

JOB JT - Message Receiving

Messages are received by activating JOB JT. When a message arrives at its final destination, the host communication process invokes the message receiving JOB JT. This job calls two routines:

- Routine ..TOT - which updates the message statistics and
- Routine ..DM - which destroys the message.

The total number of instances for JOB JT can be interpreted as the total number of messages received at JT's host site. Refer to the DSS User's Manual for more information relating to Job/External Process reports.

4.3.3 SIMULATOR OUTPUT

The DSS Simulator produces two output reports:

- Message Statistics
- ECSS Standard Reports

For a description of each report contained in this section refer to Section 2.4 of the User's Manual. For a complete sample of these reports refer to Appendix A - Level C. Examples of these reports follow:

MESSAGE STATISTIC REPORT

| TYPE | NO. OF MESSAGES | DELAY TIME MIN | MAX | AVE | STD |
|---|---|---|---|---|---|
| 1 | 27 | 3.70 | 31.26 | 17.43 | 8.50 |
| 2 | 39 | 2.70 | 32.80 | 14.64 | 7.52 |
| 3 | 26 | 1.70 | 30.10 | 15.75 | 8.58 |
| 4 | 8 | 1.40 | 6.40 | 4.00 | 1.60 |
| 5 | 33 | 1.71 | 24.11 | 13.26 | 6.76 |

SYSTEM - WIDE

| NO. OF MESSAGES | DELAY TIME MIN | MAX | AVE | STD |
|---|---|---|---|---|
| 133 | 1.40 | 32.80 | 14.44 | 8.13 |

- Refer to Section 2.4 for Report Description
- Refer to Appendix A - Level C for output sample

EXECUTION REPORT

00321

S I M U L A T I O N  S T A T I S T I C S
FROM 0. TO 50.000000

--- EXECUTION REPORT ---

| DEVICE NAME | TOTAL EXECUTION TIME | JOB NAME | JOB EXECUTION TIME | % OF DEVICE EXECUTION |
|---|---|---|---|---|
| D01A.CPU | 5.898000 | JT01 | .900000 | 15.259 |
| | | JU01 | 4.998000 | 84.741 |

--- PROCESSOR UTILIZATION REPORT ---

| DEVICE NAME | NUMBER OF ACTIVATIONS | U T I L I Z A T I O N T I M E AVERAGE | MAXIMUM | % TIME UTILIZED |
|---|---|---|---|---|
| D01A.CPU | 108 | .054611 | .102000 | 11.796 |
| D02A.CPU | 150 | .066667 | .102000 | 20.000 |
| D03A.CPU | 227 | .062432 | .102000 | 28.344 |

--- EXECUTION (READY) QUEUE REPORT ---

| DEVICE NAME | NUMBER OF ARRIVALS | W A I T F O R S E R V I C E AVERAGE | MAXIMUM | Q U E U E L E N G T H AVG | STD DEV | MAX | % TIME EMPTY |
|---|---|---|---|---|---|---|---|
| D01A.CPU | 108 | .001804 | .098000 | .004 | .062 | 1 | 99.610 |
| D02A.CPU | 150 | .001373 | .102000 | .004 | .064 | 1 | 99.588 |
| D03A.CPU | 227 | .034815 | .140539 | 158 | .413 | 2 | 86.077 |

EXAMPLES OF THE EXECUTION REPORTS - FOR DESCRIPTION OF EACH REPORT, REFER TO SECTION 2.4.

--- JOB STORE REPORT ---

| DEVICE NAME | JOB TYPE | NUMBER OF JOBS LOADED TOTAL | AVG | STD DEV | MAX | BATCH CAPACITY | PERCENTAGE UTILIZATION | JOB LOADED TIME AVERAGE | MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|---|
| D01A.CPU | S | 59 | .125 | .342 | 2 | -- | -- | .105875 | .202000 | 87.903 |
| D01A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D02A.CPU | S | 100 | .207 | .415 | 2 | -- | -- | .103350 | .205350 | 79.743 |
| D02A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D03A.CPU | S | 141 | .450 | .783 | 3 | -- | -- | .159569 | .304000 | 71.300 |
| D03A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |

--- BATCH JOB STORE QUEUE REPORT ---

| DEVICE NAME | TOTAL REQUESTS | AVG WAIT FOR LOADING | TOTAL ENQUEUED | QUEUE LENGTH AVG | STD DEV | MAX | TIME IN QUEUE AVERAGE | MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|
| D01A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D02A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D03A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D04A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D05A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D06A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D07A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D08A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D09A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |

EXAMPLES OF JOB STORE REPORTS - FOR A DESCRIPTION OF EACH REPORT, REFER TO SECTION 2.4

00321

SIMULATION STATISTICS
FROM 0. TO 50.000000

--- TRANSMISSION PATH REPORT ---

| PATH NAME | NUMBER OF TRANSMISSIONS TOTAL* | AVERAGE | STD DEV | MAXIMUM | TRANSMISSION LENGTH AVERAGE | MAXIMUM | PERCENTAGE UTILIZATION |
|---|---|---|---|---|---|---|---|
| 109B.N09.N05 | 136 | .045 | .207 | 1 | .016498 | .028128 | 4.487 |
| 109A.N09.N04 | 9 | .001 | .024 | 1 | .003278 | .003278 | .059 |
| 108B.N08.N09 | 72 | .010 | .098 | 1 | .006773 | .007443 | .975 |
| 108A.N08.N03 | 140 | .015 | .123 | 1 | .005519 | .008050 | 1.546 |

--- TRANSMISSION PATH QUEUE REPORT ---

| PATH NAME | TOTAL REQUESTS | AVG WAIT FOR TRANSMISSION | TOTAL ENQUEUED | QUEUE LENGTH AVG | STD DEV | MAX | TIME IN QUEUE AVERAGE | MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|
| 109B.N09.N05 | 136 | .000070 | 1 | .000 | .014 | 1 | .009461 | .009461 | 99.981 |
| 109A.N09.N04 | 9 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| 108B.N08.N09 | 72 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| 108A.N08.N03 | 140 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| 107B.N07.N08 | 115 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |

--- TRANSMISSION REPORT ---

| DEVICE NAME | NUMBER OF TRANSMISSIONS TOTAL | AVERAGE | STD DEV | MAXIMUM | TRANS CAPACITY | PERCENTAGE UTILIZATION | -- TRANSMISSION LENGTH -- AVERAGE | MAXIMUM | % TIME IDLE |
|---|---|---|---|---|---|---|---|---|---|
| T06A.M02 | 58 | .003 | .058 | 1 | ***** | .000 | .002887 | .003138 | 99.665 |
| T01A.M01 | 58 | .003 | .058 | 1 | ***** | .000 | .002887 | .003138 | 99.665 |
| T06B.M02 | 100 | .014 | .116 | 1 | ***** | .000 | .006799 | .012650 | 98.640 |
| T07A.M02 | 100 | .014 | .116 | 1 | ***** | .000 | .006799 | .012650 | 98.640 |
| T06C.M02 | 43 | .011 | .104 | 1 | ***** | .000 | .012700 | .012700 | 98.968 |
| T09A.M02 | 43 | .011 | .104 | 1 | ***** | .000 | .012700 | .012700 | 98.908 |
| T07B.M02 | 99 | .011 | .104 | 1 | ***** | .000 | .005507 | .012600 | 98.910 |

EXAMPLES OF TRANSMISSION REPORTS - FOR A DESCRIPTION FOR EACH REPORT, REFER TO SECTION 2.4.

-- CRITICAL DEVICE TRANSMISSION QUEUE REPORT ---

| DEVICE NAME | TOTAL ENQUEUED | QUEUE LENGTH AVG | QUEUE LENGTH STD DEV | QUEUE LENGTH MAX | TIME IN QUEUE AVERAGE | TIME IN QUEUE MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|
| T06A.M02 | 2 | .000 | .003 | 1 | .000288 | .000288 | 99.999 |
| T01A.M01 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T06B.M02 | 2 | .000 | .007 | 1 | .001112 | .001112 | 99.996 |
| T07A.M02 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T06C.M02 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T09A.M02 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T07B.M02 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T02A.M01 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |

--- TRANSMISSION RATE REPORT ---

| DEVICE NAME | TOTAL MESSAGES | -- CUMULATIVE TRANSMISSION RATE -- AVERAGE | STD DEV | MAXIMUM | TRANSMISSION RATE CAPACITY | PERCENTAGE UTILIZATION | TRANS RATE PER MESSAGE AVERAGE | MAXIMUM | % TIME IDLE |
|---|---|---|---|---|---|---|---|---|---|
| T06A.M02 | 58 | 113.9 | 1964.4 | 34000.0 | 40000.000 | .285 | 34000.0 | 34000.0 | 99.665 |
| T01A.M01 | 58 | 113.9 | 1964.4 | 34000.0 | 34000.000 | .335 | 34000.0 | 34000.0 | 99.665 |
| T06B.M02 | 100 | 543.9 | 4632.5 | 40000.0 | 40000.000 | 1.360 | 40000.0 | 40000.0 | 98.610 |
| T07A.M02 | 100 | 543.9 | 4632.5 | 40000.0 | 40000.000 | 1.360 | 40000.0 | 40000.0 | 98.610 |

EXAMPLES OF TRANSMISSION REPORTS (CONTINUED) - FOR A DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

JOB REPORT FOR JW09
FROM 0. TO 50.00

| | |
|---|---|
| TOTAL NUMBER OF INSTANCES | 139 |
| NUMBER OF COMPLETED INSTANCES | 120 |
| AVERAGE INSTANCE LENGTH | 3.905276 |
| STANDARD DEVIATION INSTANCE LENGTH | 1.776454 |
| MAXIMUM INSTANCE LENGTH | 6.988539 |
| TOTAL LENGTH OF INSTANCES | 542.833394 |

| | TIME PER INSTANCE | | | | % TOTAL |
|---|---|---|---|---|---|
| | AVERAGE | STD DEV | MAXIMUM | TOTAL | INSTANCE LENGTH |
| EXECUTION | .351050 | .131043 | .400000 | 48.796000 | 8.989 |
| TRANSMISSION | .007765 | .004096 | .012700 | 1.079360 | .199 |
| BLOCKED FOR LOADING | 0. | 0. | 0. | 0. | 0. |
| BLOCKED FOR ACTIVATION | 3.546393 | 1.766724 | 6.581096 | 492.948573 | 90.810 |
| BLOCKED FOR TRANSMISSION | .007833 | .004152 | .015922 | 1.088821 | .201 |
| BLOCKED FOR ALLOCATION | 0. | 0. | 0. | 0. | 0 |
| BLOCKED FOR STORAGE | 0. | 0. | 0. | 0. | 0. |



EXTERNAL PROCESS REPORT FOR CP01
FROM 0. TO 50.00

| | |
|---|---|
| TOTAL NUMBER OF INSTANCES | 1 |
| NUMBER OF COMPLETED INSTANCES | 0 |
| AVERAGE INSTANCE LENGTH | 50.000000 |
| STANDARD DEVIATION INSTANCE LENGTH | 0. |
| MAXIMUM INSTANCE LENGTH | 50.000000 |
| TOTAL LENGTH OF INSTANCES | 50.000000 |

| | TIME PER INSTANCE | | | | % TOTAL |
|---|---|---|---|---|---|
| | AVERAGE | STD DEV | MAXIMUM | TOTAL | INSTANCE LENGTH |
| TRANSMISSION | 0. | 0. | 0. | 0. | 0. |
| BLOCKED FOR TRANSMISSION | 0. | 0. | 0. | 0. | 0. |
| BLOCKED FOR ALLOCATION | 0. | 0. | 0. | 0. | 0. |
| BLOCKED FOR STORAGE | 0. | 0. | 0. | 0. | 0. |

EXAMPLES OF JOB/PROCESS REPORTS - FOR A DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

## 4.4 HIGH LEVEL MODEL 2

High Level Model 2 differs from High Level Model 1 in five major ways:

1. The model implements two forms of flow control procedures.
2. Messages are generated and terminated at input/output devices.
3. Messages are divided into packets before they are transmitted through the network.
4. Buffer space is explicitly modeled.
5. Packets are reassembled into messages at host sites.

### 4.4.1 MODEL ARCHITECTURE

High Level Model 2 consists of two model types:

- A model to describe the behavior of the host sites and
- A model to describe the behavior of the switching nodes.

#### 4.4.1.1 Host Sites

Each Host site consists of:

- A processor
- A variable number of internodal transmission devices (refer to High Level Model 1)
- 10 I/O devices
- 2 Channels
- 1 Buffer

Figure 4.4.1.1-1 illustrates the Host Site Configuration.

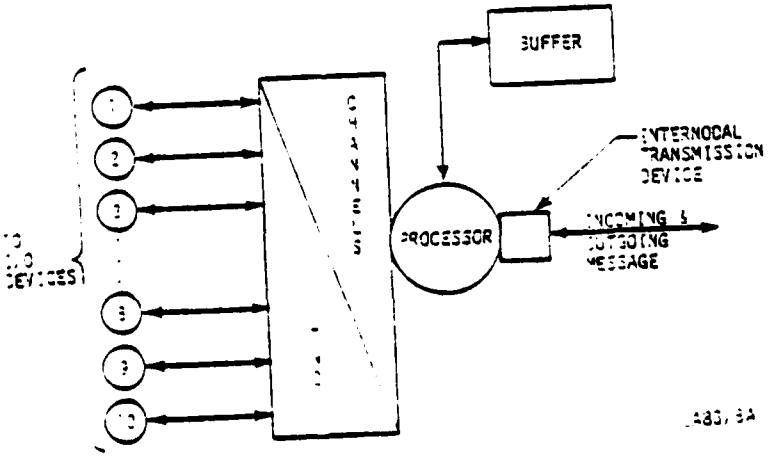



Figure 4.4.1.1-1 Host Site Configuration

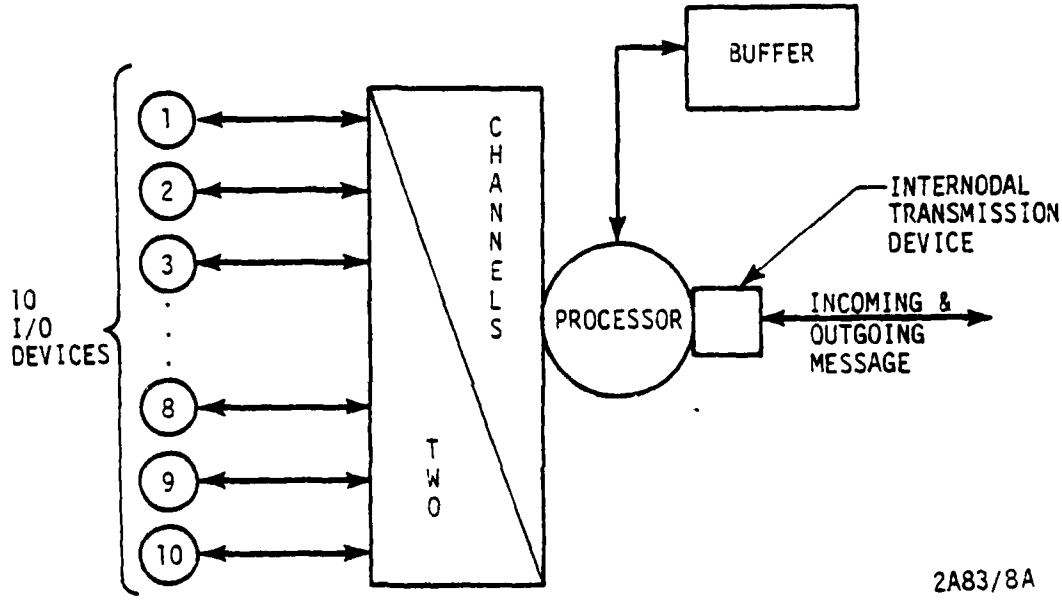



Figure 4.4.1.1-1 Host Site Configuration

The number of I/O devices and channels can be changed in the DSS High Level Model 2 host model. The Internodal device specifications are defined in the TP.FILE. Buffer specifications are changeable through the EXEC file and the System Description Section of the host model. The System Description specifications for the host site are summarized in Table 4.4.1.1-1.

4.4.1.2 Switching Nodes

Each switching node consists of:

- A Processor
- A variable number of Internodal Transmission devices (Refer to Section 4.3.1.2 Switching Nodes High Level Model 1).
- A Buffer

Figure 4.4.1.2-1 is a representation of such a switching node.

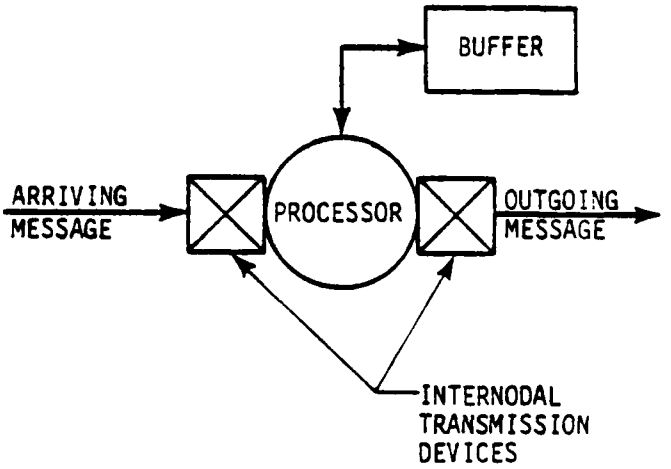



Figure 4.4.1.2-1 Switching Node Configuration (Dedicated Option)

The Processor and Buffer declarations are provided in the DSS Switching Model. The specifications for the Internodal Transmission devices are defined in the TP.FILE. Table 4.4.1.2-1 provides a summary for the System Description Section for the switching node.

Table 4.4.1.1-1 System Description for Host Site

| DEVICE NAME | ECSS DEVICE TYPE | PERFORMANCE SPECIFICATIONS |
|---|---|---|
| D01A.CPU (Processor) | EXECUTIONS ANDS JOB STORE DEVICE | 500,000 INSTRUCTIONS/SEC |
| T01A.M01 (Internodal Transmission Device) | TRANSMISSION DEVICE | 20,000 BYTES/SEC |
| D01B. CHANNELS (Channel) | TRANSMISSION DEVICE | 2,000,000 BYTES/SEC |
| D01D.BUF (Buffer) | STORAGE DEVICE | 10,000 BYTES |
| D01C.TERMINAL (Terminal) | I/O DEVICE | 9600 BYTES/SEC |
| P01A.T.PATH (Internodal Path) | PATH | CONNECT D01B. CHANNELS TO D01C TERMINALS |

Table 4.4.1.2-1 System Description for Switching Node

| DEVICE NAME | ECSS DEVICE TYPE | PERFORMANCE SPECIFICATIONS |
|---|---|---|
| D02A.CPU (Processor) | EXECUTION AND JOB STORE DEVICE | 500,000 INSTRUCTIONS/SEC |
| T02A.M01 (Internodal Transmission Device) | TRANSMISSION DEVICE | 30,000 BYTES/SEC |
| D02B.BUF (Buffer) | STORAGE DEVICE | 10,000 BYTES |
| P01A.N01.N02 (Internodal Path) | PATH | CONNECTS T01A.M01 TO T02A.M02 |

### 4.4.2 FUNCTIONAL LOGIC FLOW

High Level Model 2 simulates a store and forward packet switching network. This model incorporates packet switching and flow control procedures.

Flow control procedures are implemented by both the host and switching nodes. These techniques include:

- In the host site three limits are set by the user, they include:
  - The upper limit (MM3) on the number of unacknowledged packets that may be transmitted from a host site to the communication network. This limit is specified in the EXEC file.

  - The upper limit (MM2) on the number of messages that can be assembled at any moment. This limit is specified in the EXEC file.

  - The actual buffer size (BFH) at the host site. This value is set in the EXEC file.

- In the switching node two limits are set by the user. They include:
  - The upper limit (MM1) on the number of packets at a switching node waiting to be transferred or acknowledged. This limit is set in the EXEC file.

  - The buffer size (BFC) at the switching node. This limit is also set in the EXEC file.

There are three major functional flow logic areas within High Level Model 2. They are:

- Host site message initiation logic
- Host site message reassembly logic
- Switching node packet handling logic

The host site message initiation logic is displayed in Figure 4.4.2-1. Messages are generated at the terminal. The messages are divided into packets according to a user specified size (PKZ) and filed in the input queue. Packets may be assigned a priority (PTY). The lowest assignable priority is a "0". A packet's priority defaults to zero if no default is defined. The MESS file for High Level Model 1 can be used for High Level Model 2 if the packet size (PKZ) specified by the user is equal to a constant message length (MML). In this way, the MESS file is upwardly compatible.

The communication process verifies that the upper limit on the number of unacknowledged packets (MM3) that may be transmitted has not been reached. If this limit has been reached, the packet is denied transmission and forced to wait until an acknowledgement for one of the outstanding packets is received. If the limit has not been reached the packet is allowed to be transmitted. An acknowledgement of a user specified length (AKL) from the receiving node indicates a successful transfer of the packet. If the host does not receive this acknowledgement within a user specified time (TMH), it retransmits the packet. This cycle of waiting and retransmitting continues until a successful transfer is accomplished.

The host site message reasembly logic is illustrated in Figure 4.4.2-2. Packets arrive from adjacent nodes to be reassembled at the destination host site. Upon arrival two limiting factors, the reassembly buffer size (BFC) and the number of messages being reassembled simultaneously (MM2), are checked. If either factor exceeds its user defined limit the arriving packet is ignored and the acknowledgement to the sender node is aborted. If neither of the limits is exceeded, the communicaton process accepts the packet, allocates buffer space and sends to the forwarding node an acknowledgment of the packet's arrival. A check is made to determine if this packet is the last packet missing from the reassembled message. If it is, the reassembled message is sent to the terminals and the allocated buffer space is released. If more packets are required, the arriving packet is stored until all of the packets required to reassemble the message arrive. Single packet messages are sent directly to an I/O device and no reassembly buffer space is allocated.

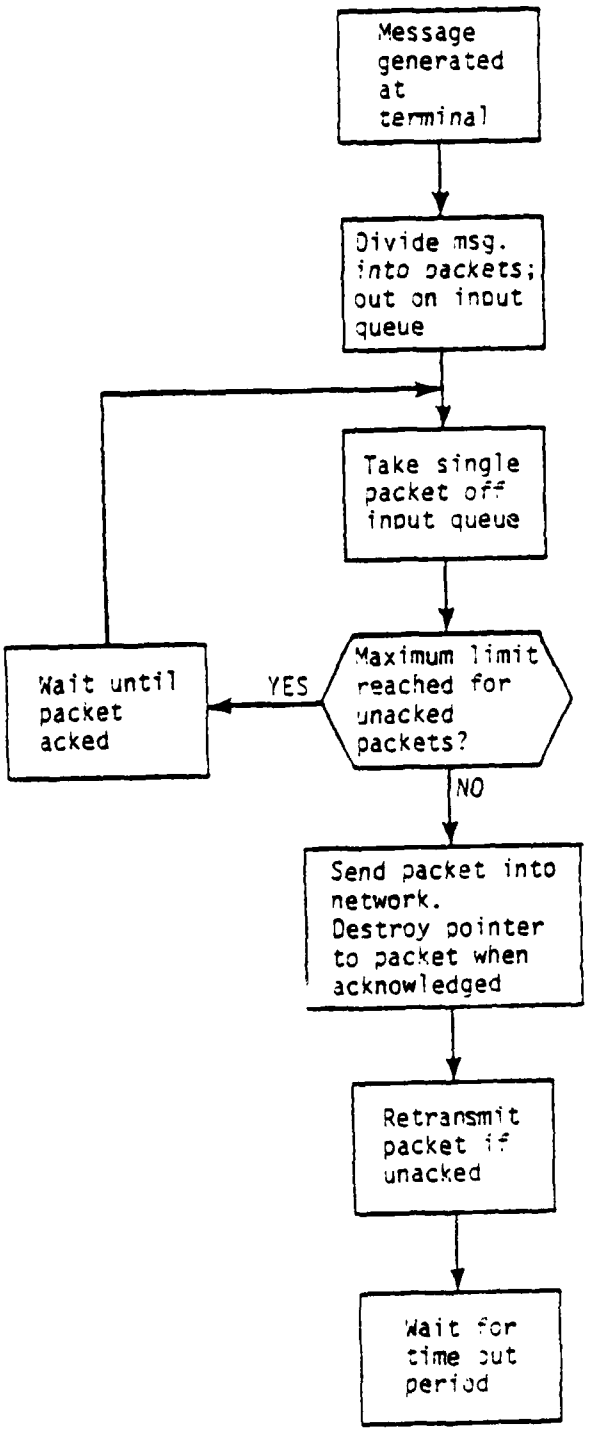
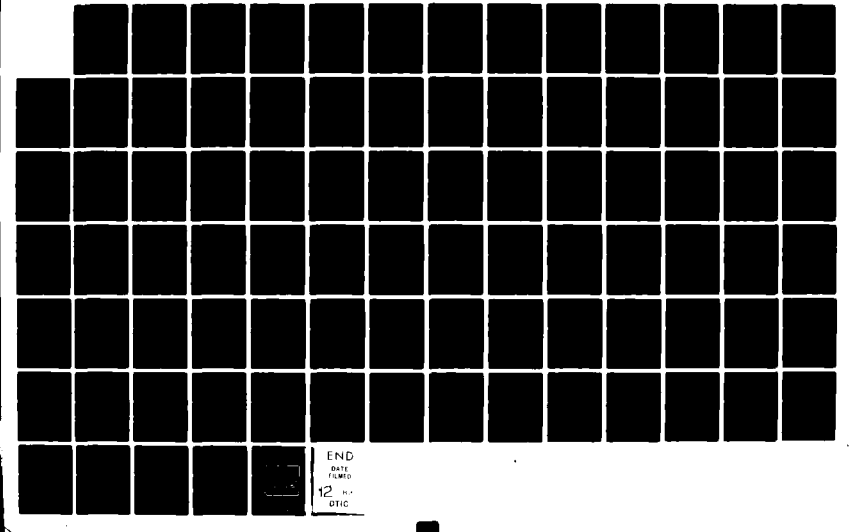



Figure 4.4.2-1 Host Site Message Initiation Logic

AD-A134 701 RADC (ROME AIR DEVELOPMENT CENTER) DDP (DISTRIBUTED DATA PROCESSING) TOPOLOGY EVALUATION(U) GENERAL ELECTRIC CO SUNNYVALE CALIF D DUBOIS AUG 83

UNCLASSIFIED RADC-TR-83-79 F30602-80-C-0267 F/G 9/2 NL

2/2

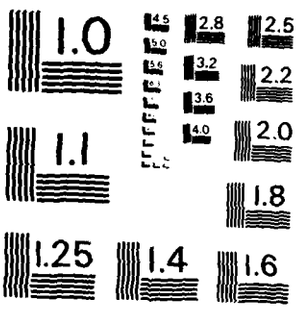


END

DATE FILMED

12 H-

DTIC



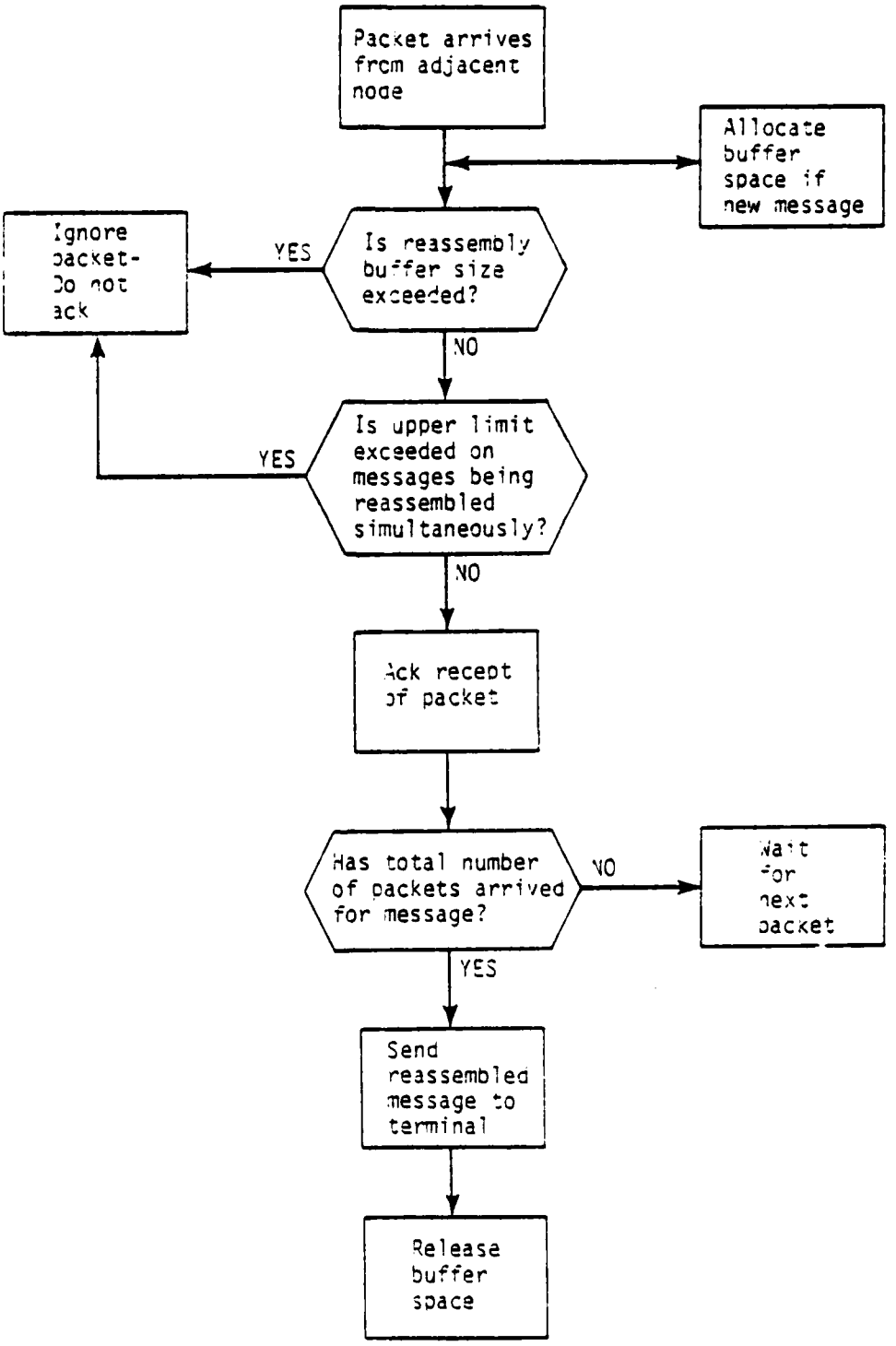



Figure 4.4.2-2 Host Site Message Reassemble Logic

The switching node packet handling logic is illustrated in Figure 4.4.2-3. Switching nodes accept packets from adjacent nodes and forward these packets to another switching node or a host site. However, the acceptance of the packet is predicated on two limiting factors:

- available buffer storage (BFC) and
- the number of packets currently at the switching node (MMI).

If either of these user defined limits are exceeded, the switching node simply ignores the packet and does not send an acknowledgement to the sending node. If neither of the limits are exceeded the switching node sends an acknowledgement to the sending node and transmits the packet to its next destination. The communication process retransmits the packet if it does not receive an acknowledgement within a user-specified time (TMC).

The communication process for High Level Model 2 provides the same type of functions as the communication process for High Level Model 1.

In addition to activating the many computer jobs, this communication process accomplishes the following:

1. Breaks messages into packets according to a user specified size (PKZ),
2. Allocates buffer space,
3. Monitors unacknowledged messages,
4. Initiates the retransmit process for all unacknowledged packets,
5. Provides total control on all packet transmissions; internodal and intranodal.

Jobs required to implement packet switching and flow control are described below:

- JOB JA - Job JA simulates the transmission of a packet from the I/O device to the CPU processor. The number of instances of this Job indicates the number of packets originating from the host site which utilize JOB JA.

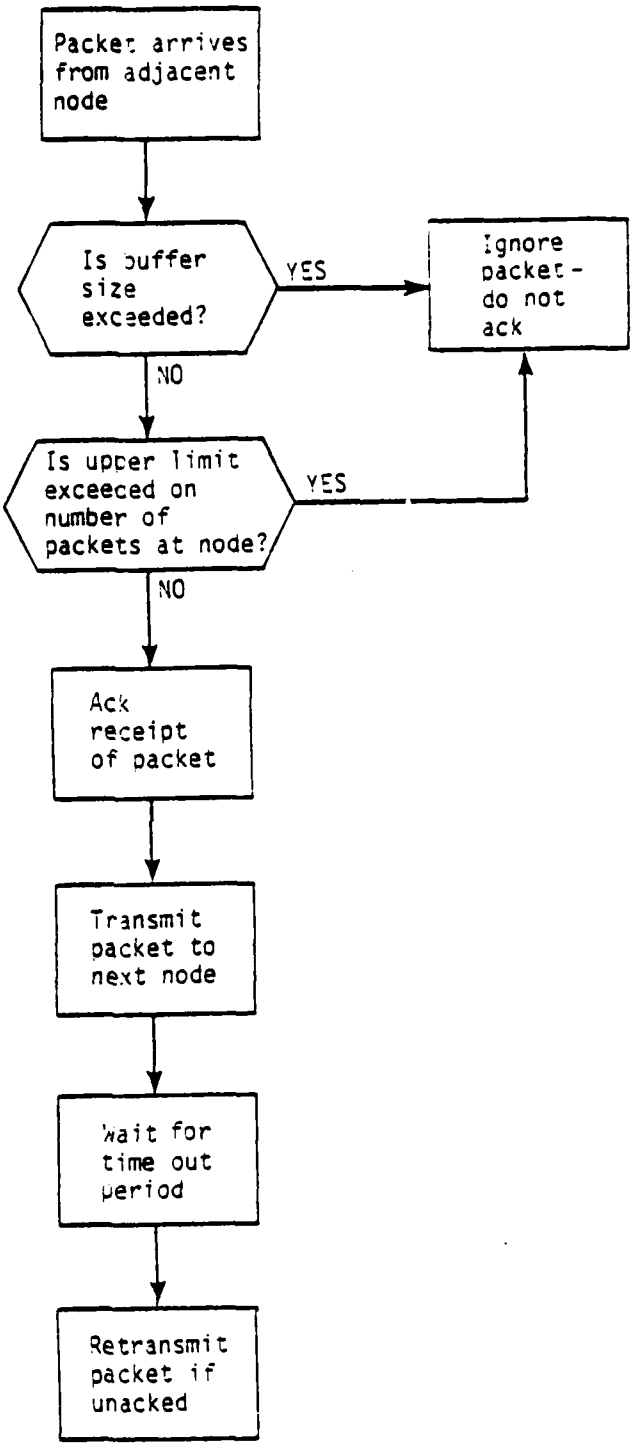



Figure 4.4.2-3 Switching Node

JOB JC - JOB JC simulates the transmission of a packet along an Internodal Path. The number of instances of this Job would indicate how many packets were injected into the network at a particular node.

JOB JU - JOB JU retransmits the packet if an acknowledgement has not been received within a user specified time. The number of instances of JOB JU indicates the number of packets sent from a particular node which were rejected by an adjacent node.

JOB AK - JOB AK is responsible for sending acknowledgements for accepted packets. This job is only operational when a "1" is indicated for "ACK" in the EXEC file. The number of instances of JOB AK indicates the number of packets accepted by a particular node.

JOB JD - JOB JD simulates the receiving and the acceptance of acknowledgements from adjacent nodes. The number of instances of this JOB indicates the number of acknowledgements received from adjacent nodes.

JOB JB - JOB JB simulates the transmission of a completely reassembled message from the CPU processor to the I/O devices. The number of instances of JOB JB indicates the total number of completely reassembled messages at a host node.

JOB PC - JOB PC is the timing mechanism that schedules the retransmit command for all unacknowledged packets. The user specifies the frequency with which this job is performed by selecting the maximum response time tolerable for all acknowledgements at the host (TMH) and switching nodes (TMC). The number of instance of this job indicates how many times a particular node assessed the status of its outgoing packets.

### 4.4.3 SIMULATOR OUTPUT

The same type of reports are generated by DSS as are produced for High Level Model 1. Complete descriptions of these reports are provided in the DSS User's Manual.

MESSAGES STATISTICS

| TYPE | NO. OF MESSAGES | DELAY TIME MIN | MAX | AVE | STD |
|---|---|---|---|---|---|
| 1 | 208 | 1.08 | 2.17 | 1.60 | .39 |
| 2 | 208 | 1.08 | 2.36 | 1.69 | .37 |
| 3 | 163 | .38 | .64 | .44 | .07 |
| 4 | 200 | .38 | .78 | .45 | .08 |

SYSTEM - WIDE

| NO. OF MESSAGES | DELAY TIME MIN | MAX | AVE | STD |
|---|---|---|---|---|
| 784 | .38 | 2.36 | 1.08 | .66 |

Message Statistics Report - For A complete Description of The Report
Refer to Section 2.4

# S I M U L A T I O N  S T A T I S T I C S

FROM 0. TO 211.900000

--- EXECUTION REPORT ---

| DEVICE NAME | TOTAL EXECUTION TIME | JOB NAME | JOB EXECUTION TIME | % OF DEVICE EXECUTION |
|---|---|---|---|---|
| D01A CPU | 8.020000 | JB01 | .690000 | 8.603 |
| | | JC01 | 2.100000 | 26.185 |
| | | AK01 | 2.080000 | 25.935 |

--- PROCESSOR UTILIZATION REPORT ---

| DEVICE NAME | NUMBER OF ACTIVATIONS | U T I L I Z A T I O N  T I M E AVERAGE | MAXIMUM | % TIME UTILIZED |
|---|---|---|---|---|
| D01A.CPU | 1394 | .005753 | .010000 | 3.785 |
| D02A.CPU | 1272 | .005660 | .010000 | 3.398 |
| D03A.CPU | 1391 | .005744 | .010000 | 3.771 |
| D04A.CPU | 1304 | .005767 | .010000 | 3.519 |

--- EXECUTION (READY) QUEUE REPORT ---

| DEVICE NAME | NUMBER OF ARRIVALS | W A I T  F O R  S E R V I C E AVERAGE | MAXIMUM | Q U E U E  L E N G T H AVG | STD DEV | MAX | % TIME EMPTY |
|---|---|---|---|---|---|---|---|
| D01A.CPU | 1394 | .002529 | .030000 | .017 | .160 | 4 | 98.745 |
| D02A.CPU | 1272 | .002184 | .022500 | .013 | .129 | 3 | 98.860 |
| D03A.CPU | 1391 | .002725 | .030000 | .018 | .173 | 3 | 98.731 |

EXECUTION REPORT - FOR A COMPLETE DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

--- JOB STORE REPORT ---

| DEVICE NAME | JOB TYPE | NUMBER OF JOBS LOADED TOTAL | NUMBER OF JOBS LOADED AVG | NUMBER OF JOBS LOADED STD DEV | NUMBER OF JOBS LOADED MAX | BATCH CAPACITY | PERCENTAGE UTILIZATION | JOB LOADED TIME AVERAGE | JOB LOADED TIME MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|---|
| D01A.CPU | S | 802 | .282 | .632 | 5 | -- | -- | .074414 | .457264 | 79.826 |
| D01A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D02A.CPU | S | 720 | .258 | .566 | 5 | -- | -- | .075848 | .206250 | 79.712 |
| D02A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D03A.CPU | S | 799 | 1.888 | 1.644 | 6 | -- | -- | .500603 | 1.687083 | 29.814 |
| D03A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |

--- BATCH JOB STORE QUEUE REPORT ---

| DEVICE NAME | TOTAL REQUESTS | AVG WAIT FOR LOADING | TOTAL ENQUEUED | QUEUE LENGTH AVG | QUEUE LENGTH STD DEV | QUEUE LENGTH MAX | TIME IN QUEUE AVERAGE | TIME IN QUEUE MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|
| D01A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D02A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D03A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D04A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D05A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D06A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D07A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D08A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D09A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |

JOB STORE REPORT - FOR A COMPLETE DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

--- TRANSMISSION PATH REPORT ---

| PATH NAME | NUMBER OF TRANSMISSIONS TOTAL* | AVERAGE | STD DEV | MAXIMUM | TRANSMISSION LENGTH AVERAGE | MAXIMUM | PERCENTAGE UTILIZATION |
|---|---|---|---|---|---|---|---|
| P01A.T.PATH | 184 | .146 | .362 | 2 | .167572 | .177083 | 14.197 |
| P03A.T.PATH | 174 | .166 | .377 | 2 | .202347 | .270833 | 16.421 |
| P02A.T.PATH | 184 | .146 | .369 | 2 | .167572 | .177083 | 13.962 |
| P01A.T.PATH | 174 | .166 | .411 | 2 | .201688 | .270833 | 15.011 |
| 103A.N03.N09 | 834 | .921 | .269 | 1 | .234109 | .625000 | 92.141 |
| 102A.N02.N08 | 736 | .081 | .272 | 1 | .023234 | .062500 | 8.070 |

--- TRANSMISSION PATH QUEUE REPORT ---

| PATH NAME | TOTAL REQUESTS | AVG WAIT FOR TRANSMISSION | TOTAL ENQUEUED | QUEUE LENGTH AVG | STD DEV | MAX | TIME IN QUEUE AVERAGE | MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|
| P04A.T.PATH | 184 | .001619 | 2 | .001 | .037 | 1 | .148958 | .157083 | 99.859 |
| P03A.T.PATH | 174 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| P02A.T.PATH | 184 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| P01A.T.PATH | 174 | .001557 | 1 | .001 | .036 | 1 | .270833 | .270833 | 99.872 |
| 103A.N03.N09 | 838 | .685983 | 787 | 2.713 | 1.441 | 6 | .730436 | 1.607083 | 11.581 |
| 102A.N02.N08 | 736 | .009920 | 252 | .034 | .204 | 3 | .028973 | .092681 | 96.932 |
| 104A.N04.N05 | 736 | .007953 | 291 | .028 | .165 | 3 | .020115 | .089167 | 97.529 |
| 101A.N01.N06 | 836 | .005941 | 257 | .023 | .155 | 2 | .019326 | .045217 | 97.706 |
| 108A.N08.N09 | 840 | .060738 | 525 | .241 | .567 | 3 | .097181 | .282917 | 81.386 |
| 107C.N07.N08 | 1576 | .090402 | 1274 | .672 | 1.080 | 7 | .111832 | .469167 | 62.278 |

--- TRANSMISSION REPORT ---

| DEVICE NAME | NUMBER OF TRANSMISSIONS TOTAL | AVERAGE | STD DEV | MAXIMUM | TRANS CAPACITY | PERCENTAGE UTILIZATION | -- TRANSMISSION LENGTH -- AVERAGE | MAXIMUM | % TIME IDLE |
|---|---|---|---|---|---|---|---|---|---|
| 105A.M02 | 736 | .227 | .419 | 1 | ***** | .000 | .065489 | .091667 | 77.253 |
| 107B.M02 | 837 | .259 | .438 | 1 | ***** | .000 | .065578 | .091667 | 74.097 |
| D01C.TERMINAL 1 | 21 | .020 | .141 | 1 | ***** | .000 | .205357 | .270833 | 97.965 |
| D01C.TERMINAL 2 | 19 | .018 | .134 | 1 | ***** | .000 | .204496 | .270833 | 98.166 |
| D01C.TERMINAL 3 | 17 | .016 | .125 | 1 | ***** | .000 | .196691 | .270833 | 98.422 |
| D01C.TERMINAL 4 | 27 | .028 | .165 | 1 | ***** | .000 | .219907 | .270833 | 97.198 |
| D01C.TERMINAL 5 | 18 | .018 | .131 | 1 | ***** | .000 | .207176 | .270833 | 98.240 |
| D01C.TERMINAL 6 | 13 | .013 | .115 | 1 | ***** | .000 | .217949 | .270833 | 98.663 |

TRANSMISSION REPORT - FOR A COMPLETE DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

--- CRITICAL DEVICE TRANSMISSION QUEUE REPORT ---

| DEVICE NAME | TOTAL ENQUEUED | QUEUE LENGTH AVG | QUEUE LENGTH STD DEV | QUEUE LENGTH MAX | TIME IN QUEUE AVERAGE | TIME IN QUEUE MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|
| T07A.M02 | 563 | .162 | .455 | 3 | .060907 | .205000 | 86.516 |
| T05A.M02 | 0 | 0. | 0 | 0 | 0. | 0. | 100.000 |
| T07B.M02 | 521 | .134 | .365 | 2 | .054439 | .140000 | 87.472 |
| T06A.M02 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T07C.M02 | 1274 | .672 | 1.080 | 7 | .111832 | .469167 | 62.278 |
| T08A.M02 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T08B.M02 | 525 | .241 | .567 | 3 | .097181 | .282917 | 81.386 |
| T09A.M02 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T01A.M01 | 257 | .023 | .155 | 2 | .019326 | .045217 | 97.706 |

--- TRANSMISSION RATE REPORT ---

| DEVICE NAME | TOTAL MESSAGES | CUMULATIVE TRANSMISSION RATE AVERAGE | CUMULATIVE TRANSMISSION RATE STD DEV | CUMULATIVE TRANSMISSION RATE MAXIMUM | TRANSMISSION RATE CAPACITY | PERCENTAGE UTILIZATION | TRANS RATE PER MESSAGE AVERAGE | TRANS RATE PER MESSAGE MAXIMUM | % TIME IDLE |
|---|---|---|---|---|---|---|---|---|---|
| T07A.M02 | 736 | 6824.0 | 12575.9 | 30000.0 | 30000.000 | 22.747 | 30000.0 | 30000.0 | 77.253 |
| T05A.M02 | 736 | 6824.0 | 12575.9 | 30000.0 | 34000.000 | 20.071 | 30000.0 | 30000.0 | 77.253 |
| T07B.M02 | 837 | 7771.0 | 13143.1 | 30000.0 | 30000.000 | 25.903 | 30000.0 | 30000.0 | 74.097 |
| T06A.M02 | 837 | 7771.0 | 13143.1 | 30000.0 | 34000.000 | 22.856 | 30000.0 | 30000.0 | 74.097 |
| T07C.M02 | 1575 | 16837.8 | 14887.0 | 30000.0 | 30000.000 | 56.126 | 30000.0 | 30000.0 | 43.874 |
| T08A.M02 | 1575 | 16837.8 | 14887.0 | 30000.0 | 36000.000 | 46.772 | 30000.0 | 30000.0 | 43.874 |
| T08B.M02 | 840 | 11359.0 | 14551.4 | 30000.0 | 30000.000 | 37.863 | 30000.0 | 30000.0 | 62.137 |
| T09A.M02 | 840 | 11359.0 | 14551.4 | 30000.0 | 36000.000 | 61.553 | 30000.0 | 30000.0 | 62.137 |
| T01A.M01 | 836 | 1842.4 | 6243.4 | 23000.0 | 30000.000 | 6.141 | 23000.0 | 23000.0 | 91.990 |

TRANSMISSION REPORT (CONTINUED) - FOR A COMPLETE DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

--- STORAGE REPORT ---

| DEVICE NAME | TOTAL REQUESTS | STORAGE UTILIZATION AVERAGE | STD DEV | MAXIMUM | STORAGE CAPACITY | PERCENTAGE UTILIZATION | REQUEST SIZE AVERAGE | MAXIMUM | % TIME VACANT |
|---|---|---|---|---|---|---|---|---|---|
| D01D.BUF | 70 | 860.153 | 1576.614 | 3750.000 | 10000 | 8.602 | 3750.0 | 3750.0 | 77.063 |
| D02D.BUF | 100 | 71.436 | 416.516 | 2500.000 | 10000 | .714 | 2500.0 | 2500.0 | 97.143 |
| D03D.BUF | 104 | 160.011 | 611.901 | 2500.000 | 10000 | 1.600 | 2500.0 | 2500.0 | 93.600 |
| D04D.BUF | 84 | 24.769 | 247.605 | 2500.000 | 10000 | .248 | 2500.0 | 2500.0 | 99.009 |
| D05B.BUF | 368 | 260.846 | 596.854 | 3200.000 | 10000 | 2.608 | 804.3 | 1250.0 | 81.643 |
| D06B.BUF | 418 | 253.363 | 538.315 | 2750.000 | 10000 | 2.534 | 809.0 | 1250.0 | 78.960 |
| D07B.BUF | 788 | 901.021 | 1084.740 | 5500.000 | 10000 | 9.010 | 806.5 | 1250.0 | 46.458 |
| D08B.BUF | 788 | 830.705 | 944.686 | 4550.000 | 10000 | 8.307 | 806.5 | 1250.0 | 45.014 |
| D09B.BUF | 420 | 1414.659 | 9[illegible].131 | 4350.000 | 10000 | 14.147 | 808.3 | 1250.0 | 13.277 |



--- STORAGE QUEUE REPORT ---

| DEVICE NAME | TOTAL REQUESTS | AVG WAIT FOR SERVICE | TOTAL ENQUEUED | QUEUE LENGTH AVG | STD DEV | MAX | TIME IN QUEUE AVERAGE | MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|
| D01D.BUF | 70 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D02D.BUF | 100 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D03D.BUF | 104 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D04D.BUF | 84 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D05B.BUF | 368 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D06B.BUF | 418 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D07B.BUF | 788 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D08B.BUF | 788 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D09B.BUF | 420 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |

STORAGE REPORT - FOR A COMPLETE DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

JOB REPORT FOR JD09
FROM 0. TO 211.90

| | |
|---|---|
| TOTAL NUMBER OF INSTANCES | 417 |
| NUMBER OF COMPLETED INSTANCES | 417 |
| AVERAGE INSTANCE LENGTH | .010022 |
| STANDARD DEVIATION INSTANCE LENGTH | .000448 |
| MAXIMUM INSTANCE LENGTH | .019166 |
| TOTAL LENGTH OF INSTANCES | 4.179166 |

| | TIME PER INSTANCE | | | | % TOTAL |
|---|---|---|---|---|---|
| | AVERAGE | STD DEV | MAXIMUM | TOTAL | INSTANCE LENGTH |
| EXECUTION | .010000 | -.000000 | .010000 | 4.170000 | 99.781 |
| TRANSMISSION | 0. | 0. | 0. | 0. | 0. |
| BLOCKED FOR LOADING | 0. | 0. | 0. | 0. | 0. |
| BLOCKED FOR ACTIVATION | .000022 | .000448 | .009166 | .009166 | .219 |
| BLOCKED FOR TRANSMISSION | 0. | 0. | 0. | 0. | 0 |
| BLOCKED FOR ALLOCATION | 0. | 0. | 0. | 0. | 0. |
| BLOCKED FOR STORAGE | 0. | 0. | 0. | 0. | 0. |



EXTERNAL PROCESS REPORT FOR CP01
FROM 0. TO 211.90

| | |
|---|---|
| TOTAL NUMBER OF INSTANCES | 1 |
| NUMBER OF COMPLETED INSTANCES | 0 |
| AVERAGE INSTANCE LENGTH | 211.900000 |
| STANDARD DEVIATION INSTANCE LENGTH | -.000000 |
| MAXIMUM INSTANCE LENGTH | 211.900000 |
| TOTAL LENGTH OF INSTANCES | 211.900000 |

| | TIME PER INSTANCE | | | | % TOTAL |
|---|---|---|---|---|---|
| | AVERAGE | STD DEV | MAXIMUM | TOTAL | INSTANCE LENGTH |
| TRANSMISSION | 0. | 0. | 0. | 0. | 0. |
| BLOCKED FOR TRANSMISSION | 0. | 0. | [illegible] | [illegible] | 0. |
| BLOCKED FOR ALLOCATION | 0. | 0. | [illegible] | | 0. |
| BLOCKED FOR STORAGE | 0. | 0. | | [illegible] | 0. |

## 4.5 HIGH LEVEL MODEL 3

High Level Model 3 is different from High Level Model 2 in that it uses a non-deterministic routing table. The "C" level example of this model illustrates the upward compatibility of the DSS input files. For example, the simulation model for Level C consists of several files from High Level Model 1 and 2. No new files are required.

### 4.5.1 ADAPTIVE ROUTING

Routing algorithms may generally be classified as either fixed or adaptive. With fixed routing algorithms the next node to which to send a message is pre-determined. In adaptive routing the procedure to forward packets is sensitive to the changing state of the network. They are generally variants of shortest path algorithms that route packets from a source node to a destination over a path of least cost. The specific cost function used in a particular network may vary widely. Some costs are functions of channel capacity, link or transmission line capacity, number of hops from source to destination, error rates over given paths, expected queue lengths at various nodes and so on. Beside these differences in link cost functions, routing techniques tend to differ in implementation and the place at which the algorithms are run. They may be actually located in a single node or the decision making functions may be dispersed throughout the network. How frequently the routing tables are updated, routing decision overhead and the number of parameters the adaptive procedures are sensitive to are also distinguishing characteristics. However, a high level view of these adaptive techniques suggests these overriding characteristics: multiple paths may exist from source to destination and these paths change over time. That is, a message going from node A to node B might go by way of path X at time T and by path Y at a later time. If measured over time a certain percentage of the total number of messages going from A to B will choose path X over path Y. It is this average behavior that is approximated by a new type of routing table that is incorporated in HLM #3.

### 4.5.2 SIMULATOR OUTPUT

DSS produces the same type of reports for High Level Model 3 as it provides for High Level Models 1 and 2. Refer to the DSS User's Manual for a complete description of each report. For complete examples of each report refer to Appendix C.

MESSAGES STATISTICS

| TYPE | NO. OF MESSAGES | DELAY TIME MIN | MAX | AVE | STD |
|---|---|---|---|---|---|
| 1 | 199 | .08 | .28 | .16 | .08 |
| 2 | 188 | .09 | .18 | .12 | .08 |
| 3 | 165 | .06 | .22 | .10 | .05 |
| 4 | 40 | .17 | .22 | .19 | .02 |
| 5 | 133 | .08 | .15 | .09 | .02 |

SYSTEM - WIDE

| NO. OF MESSAGES | DELAY TIME MIN | MAX | AVE | STD |
|---|---|---|---|---|
| 720 | .06 | .28 | .12 | .06 |

Message Statistics Report - For A complete Description of The Report Refer to Section 2.4

--- EXECUTION REPORT ---

| DEVICE NAME | TOTAL EXECUTION TIME | JOB NAME | JOB EXECUTION TIME | % OF DEVICE EXECUTION |
|---|---|---|---|---|
| D01A.CPU | 4.180000 | JB01 | .4900000 | 10.987 |
| | | JC01 | 1.0000000 | 22.321 |
| | | AK01 | .9900000 | 22.099 |
| | | JA01 | 1.0000000 | 22.321 |
| | | JD01 | 1.0000000 | 22.321 |

--- PROCESSOR UTILIZATION REPORT

| DEVICE NAME | NUMBER OF ACTIVATIONS | UTILIZATION TIME AVERAGE | UTILIZATION TIME MAXIMUM | % TIME UTILIZED |
|---|---|---|---|---|
| D01A.CPU | 795 | .005635 | .010000 | 4.458 |
| D02A.CPU | 1475 | .005417 | .010000 | 7.951 |
| D03A.CPU | 2076 | .005644 | .010000 | 11.454 |
| D04A.CPU | 160 | .006250 | .010000 | .995 |

--- EXECUTION (READY) QUEUE REPORT ---

| DEVICE NAME | NUMBER OF ARRIVALS | WAIT FOR SERVICE AVERAGE | WAIT FOR SERVICE MAXIMUM | QUEUE LENGTH AVG | QUEUE LENGTH STD DEV | QUEUE LENGTH MAX | % TIME EMPTY |
|---|---|---|---|---|---|---|---|
| D01A.CPU | 795 | .001279 | .010000 | .010 | .120 | 2 | 99.205 |
| D02A.CPU | 1475 | .003291 | .028162 | .048 | .289 | 3 | 95.700 |
| D03A.CPU | 2076 | .002635 | .023301 | .054 | .264 | 4 | 95.388 |

EXECUTION REPORTS - FOR A COMPLETE DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

--- JOB STORE REPORT ---

| DEVICE NAME | JOB TYPE | NUMBER OF JOBS LOADED TOTAL | AVG | STD DEV | MAX | BATCH CAPACITY | PERCENTAGE UTILIZATION | JOB LOADED TIME AVERAGE | MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|---|
| D01A.CPU | S | 448 | .180 | .425 | 3 | -- | -- | .040316 | .228333 | 83.350 |
| D01A.CPU | B | 0 | 0. | 0 | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D02A.CPU | S | 799 | .277 | .650 | 5 | -- | -- | .034802 | .228161 | 80.905 |
| D02A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D03A.CPU | S | 1151 | .466 | .805 | 6 | -- | -- | .046681 | .242180 | 63.475 |
| D03A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D01A.CPU | S | 100 | .088 | .334 | 2 | -- | -- | .033078 | .218333 | 92.826 |
| D01A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D05A.CPU | S | 925 | .305 | .588 | 5 | -- | -- | .033130 | .233381 | 75.372 |
| D05A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D06A.CPU | S | 1147 | .259 | .685 | 4 | -- | -- | .022712 | .063601 | 85.060 |
| D06A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D07A.CPU | S | 1644 | .376 | .852 | 7 | -- | -- | .023010 | .092033 | 75.652 |
| D07A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D08A.CPU | S | 1494 | .358 | .872 | 6 | -- | -- | .024086 | .090100 | 81.012 |
| D08A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |
| D09A.CPU | S | 1068 | .268 | .736 | 6 | -- | -- | .025170 | .096988 | 85.205 |
| D09A.CPU | B | 0 | 0. | 0. | 0 | ***** | 0. | 0. | 0. | 100.000 |

--- BATCH JOB STORE QUEUE REPORT ---

| DEVICE NAME | TOTAL REQUESTS | AVG WAIT FOR LOADING | TOTAL ENQUEUED | QUEUE LENGTH AVG | STD DEV | MAX | TIME IN QUEUE AVERAGE | MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|
| D01A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D02A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D03A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D04A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D05A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D06A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D07A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D08A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D09A.CPU | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100 000 |

JOB STORE REPORT - FOR A COMPLETE DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

--- TRANSMISSION PATH REPORT ---

| PATH NAME | NUMBER OF TRANSMISSIONS TOTAL* | AVERAGE | STD DEV | MAXIMUM | TRANSMISSION LENGTH AVERAGE | MAXIMUM | PERCENTAGE UTILIZATION |
|---|---|---|---|---|---|---|---|
| P05A.T.PATH | 336 | .129 | .357 | 2 | .038476 | .208833 | 12.094 |
| P04A.T.PATH | 20 | .041 | .199 | 1 | .208996 | .208333 | 4.146 |
| P03A.T.PATH | 428 | .201 | .432 | 2 | .047276 | .208333 | 18.855 |
| P02A.T.PATH | 328 | .122 | .336 | 2 | .037436 | .208333 | 11.935 |
| P01A.T.PATH | 149 | .112 | .316 | 1 | .075353 | .208333 | 11.247 |

--- TRANSMISSION PATH QUEUE REPORT ---

| PATH NAME | TOTAL REQUESTS | AVG WAIT FOR TRANSMISSION | TOTAL ENQUEUED | QUEUE LENGTH AVG | STD DEV | MAX | TIME IN QUEUE AVERAGE | MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|
| P05A.T.PATH | 336 | .001573 | 6 | .005 | .072 | 1 | .003111 | .10000 | 99.474 |
| P01A.T.PATH | 20 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| P03A.T.PATH | 428 | .001555 | 14 | .007 | .087 | 2 | .04754 | .034120 | 99.309 |
| P02A.T.PATH | 328 | .000426 | 2 | .001 | .067 | 1 | .06328 | .124738 | 99.501 |
| P01A.T.PATH | 149 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |



--- TRANSMISSION REPORT ---

| DEVICE NAME | NUMBER OF TRANSMISSIONS TOTAL | AVERAGE | STD DEV | MAXIMUM | TRANS CAPACITY | PERCENTAGE UTILIZATION | -- TRANSMISSION LENGTH -- AVERAGE | MAXIMUM | % TIME IDLE |
|---|---|---|---|---|---|---|---|---|---|
| T06A.M02 | 398 | .040 | .195 | 1 | ***** | .000 | .010042 | .036815 | 90.023 |
| T01A.M01 | 398 | .040 | .195 | 1 | ***** | .000 | .010042 | .036815 | 96.023 |
| T06B.M02 | 766 | .055 | .228 | 1 | ***** | .000 | .007210 | .081400 | 94.475 |
| D01C.TERMINAL 1 | 17 | .012 | .107 | 1 | ***** | .000 | .068995 | .208833 | 98.833 |
| D01C.TERMINAL 2 | 18 | .014 | .116 | 1 | ***** | .000 | .076736 | .208833 | 98.625 |
| D01C.TERMINAL 3 | 10 | .007 | .033 | 1 | ***** | .000 | .070156 | .208833 | 99.302 |
| D01C.TERMINAL 4 | 12 | .009 | .095 | 1 | ***** | .000 | .076736 | .208833 | 99.084 |

TRANSMISSION REPORTS - FOR A COMPLETE DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

---- CRITICAL DEVICE TRANSMISSION QUEUE REPORT ---

| DEVICE NAME | TOTAL ENQUEUED | QUEUE LENGTH AVG | QUEUE LENGTH STD DEV | QUEUE LENGTH MAX | TIME IN QUEUE AVERAGE | TIME IN QUEUE MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|
| T06A.M02 | 100 | .014 | .118 | 1 | .014241 | .016815 | 98.583 |
| T01A.M01 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T06B.M02 | 109 | .009 | .095 | 1 | .008428 | .014400 | 99.086 |
| T07A.M02 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T06C.M02 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T09A.M02 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| T07B.M02 | 46 | .007 | .095 | 2 | .016171 | .080200 | 99.015 |
| T02A.M01 | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |

--- TRANSMISSION RATE REPORT ---

| DEVICE NAME | TOTAL MESSAGES | CUMULATIVE TRANSMISSION RATE - AVERAGE | CUMULATIVE TRANSMISSION RATE - STD DEV | CUMULATIVE TRANSMISSION RATE - MAXIMUM | TRANSMISSION RATE CAPACITY | PERCENTAGE UTILIZATION | TRANS RATE PER MESSAGE AVERAGE | TRANS RATE PER MESSAGE MAXIMUM | % TIME IDLE |
|---|---|---|---|---|---|---|---|---|---|
| T06A.M02 | 398 | 1352.3 | 6644.6 | 34000.0 | 40000.000 | 3.684 | 34000.0 | 34000.0 | 96.026 |
| T01A.M01 | 398 | 1352.3 | 6644.6 | 34000.0 | 34000.000 | 3.677 | 34000.0 | 34000.0 | 96.026 |
| T06B.M02 | 766 | 2210.2 | 9139.1 | 40000.0 | 40000.000 | 5.525 | 40000.0 | 40000.0 | 94.475 |
| T07A.M02 | 766 | 2210.2 | 9139.1 | 40000.0 | 40000.000 | 5.525 | 40000.0 | 40000.0 | 94.475 |
| T06B.M02 | 366 | 845.2 | 5752.7 | 40000.0 | 40000.000 | 4.113 | 40000.0 | 40000.0 | 97.837 |
| T05A.M02 | 366 | 845.2 | 5752.7 | 40000.0 | 40000.000 | 1.818 | 40000.0 | 40000.0 | 97.837 |
| T07B.M02 | 696 | 1633.6 | 7916.9 | 40000.0 | 40000.000 | 4.094 | 40000.0 | 40000.0 | 98.918 |
| T03A.M01 | 696 | 1633.6 | 7916.9 | 40000.0 | 40000.000 | 4.094 | 40000.0 | 40000.0 | 98.918 |

TRANSMISSION REPORT (CONTINUED) - FOR A COMPLETE DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

--- STORAGE REPORT ---

| DEVICE NAME | TOTAL REQUESTS | STORAGE AVERAGE | UTILIZATION STD DEV | UTILIZATION MAXIMUM | STORAGE CAPACITY | PERCENTAGE UTILIZATION | REQUEST SIZE AVERAGE | REQUEST SIZE MAXIMUM | % TIME VACANT |
|---|---|---|---|---|---|---|---|---|---|
| D01D.BUF | 50 | 29.477 | 209.860 | 2500.000 | 10000 | .758 | 2500.0 | 2600.0 | 99.841 |
| D02D.BUF | 0 | 0. | 0. | 0. | 10000 | 0. | 0. | 0. | 100.000 |
| D03D.BUF | 20 | 15.807 | 198.159 | 2500.000 | 10000 | .158 | 2500.0 | 2600.0 | 95.026 |
| D04D.BUF | 0 | 0. | 0. | 0. | 10000 | 0. | 0. | 0. | 100.000 |
| D05D.BUF | 20 | 22.035 | 203.670 | 2500.000 | 10000 | .229 | 2500.0 | 2600.0 | 99.119 |
| D06B.BUF | 388 | 119.458 | 391.520 | 2110.000 | 10000 | 1.195 | 443.8 | 1250.0 | 82.432 |
| D07B.BUF | 546 | 164.917 | 482.675 | 4000.000 | 10000 | 1.819 | 416.8 | 1250.0 | 76.218 |
| D08B.BUF | 498 | 152.709 | 513.505 | 4200.000 | 10000 | 1.527 | 343.3 | 1250.0 | 78.737 |
| D09B.BUF | 356 | 99.676 | 333.652 | 2200.000 | 10000 | .957 | 353.2 | 1250.0 | 82.737 |

--- STORAGE QUEUE REPORT ---

| DEVICE NAME | TOTAL REQUESTS | AVG WAIT FOR SERVICE | TOTAL ENQUEUED | QUEUE LENGTH AVG | QUEUE LENGTH STD DEV | QUEUE LENGTH MAX | TIME IN QUEUE AVERAGE | TIME IN QUEUE MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|
| D01D.BUF | 50 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D02D.BUF | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D03D.BUF | 20 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D04D.BUF | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D05D.BUF | 20 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D06B.BUF | 333 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D07B.BUF | 543 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D08B.BUF | 498 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| D09B.BUF | 355 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |

STORAGE REPORTS - FOR COMPLETE DESCRIPTION OF THESE REPORTS, REFER TO SECTION 2.4.

# SECTION 5
# DETAILED MODELS OF COMPUTER NETWORKS

## 5.1 OVERVIEW

This section gives a summary description of the design and rationale for each of the Detailed Models for Phase II of the RADC DDP Topology Evaluation Project. (Succeeding sections describe each of these models at greater length.)

The three detailed models for phase II are:

- Communication Protocol (CP) Detailed Model
- Reliability/Availability (R/A) Detailed Model
- Distributed Database (DB) Detailed Model

We have been concerned during both phases of the DDP project with two primary tasks:

1) Development of high level (Phase I) and detailed level (Phase II) models of distributed data processing systems that would have wide applicability;
2) Development of a modeling tool which would facilitate the building of simulators that could be put into an expandable model library for later use and enhancement. This tool is called DSS.

In light of these two primary tasks, the design and rationale for the detailed models must serve the functions outlined above. In particular, DSS should be able to model DDP Systems incorporating a great deal of variability in the level of detail required. As a reference point the International Standards Organization's (ISO) seven layer architectural design was used. The highest level is the application level which includes distributed data base management systems (DDBM). The lowest layer of the ISO Model that has any significant impact on performance is the data link layer. (DSS incorporates a propagation delay factor where that is appropriate, for instance satellite links or long

lines, which is on the physical layer.) Accordingly, we chose two models that incorporate either end of the ISO Reference Model: the DB System Model for the application level and the CP Model for the data link layer. Each of these models incorporates all of the seven layers of the ISO Reference Model (for example, each contains a routing mechanism which is part of the network layer) but special attention is paid to the application layer in the DB Model and the data link layer in the CP Model. Figure 5.1-1 illustrates the ISO Reference Model's layered Network Architecture.

But this is only one of the reasons for having a DB Model and CP Model. The range of applicability is also important. One of the most important applications in distributed data processing technology is DDBM Systems. Almost any computer network will incorporate a DDBM System. Therefore a DDBM Model is an important test for evaluating the capabilities of DSS.

One of the most fundamental ways in which a distributed system differs from the traditional centralized systems is in the area of communication protocols. DSS may be viewed as an extension of ECSS. ECSS was designed by the Rand Corporation in 1975 to make the modeling of centralized systems easier. DSS uses ECSS as a building block and adds, among other things, an Inter-nodal Communication Facility. This facility, along with the ECSS transmission manager, allows the user to model communication protocols effectively and easily. Because communication protocols are so important to the performance of computer networks and because modeling this aspect of DDP Systems goes beyond the traditional world view of ECSS we have chosen to include the CP Model. The particular communication protocol that has been modeled is the X.25 interface for packet switched networks which includes the High Level Data Link Control (HDLC) protocol on the data link level of the ISO Model. This is the most widely used communication protocol and has been adopted by several international organizations, including ISO, and nationally by ANSI.

The Reliability/Availability (R/A) Model is included for several reasons. One of the main rationales for the development of DDP technology is the increase in the reliability/availability of the system that can be realized from a well

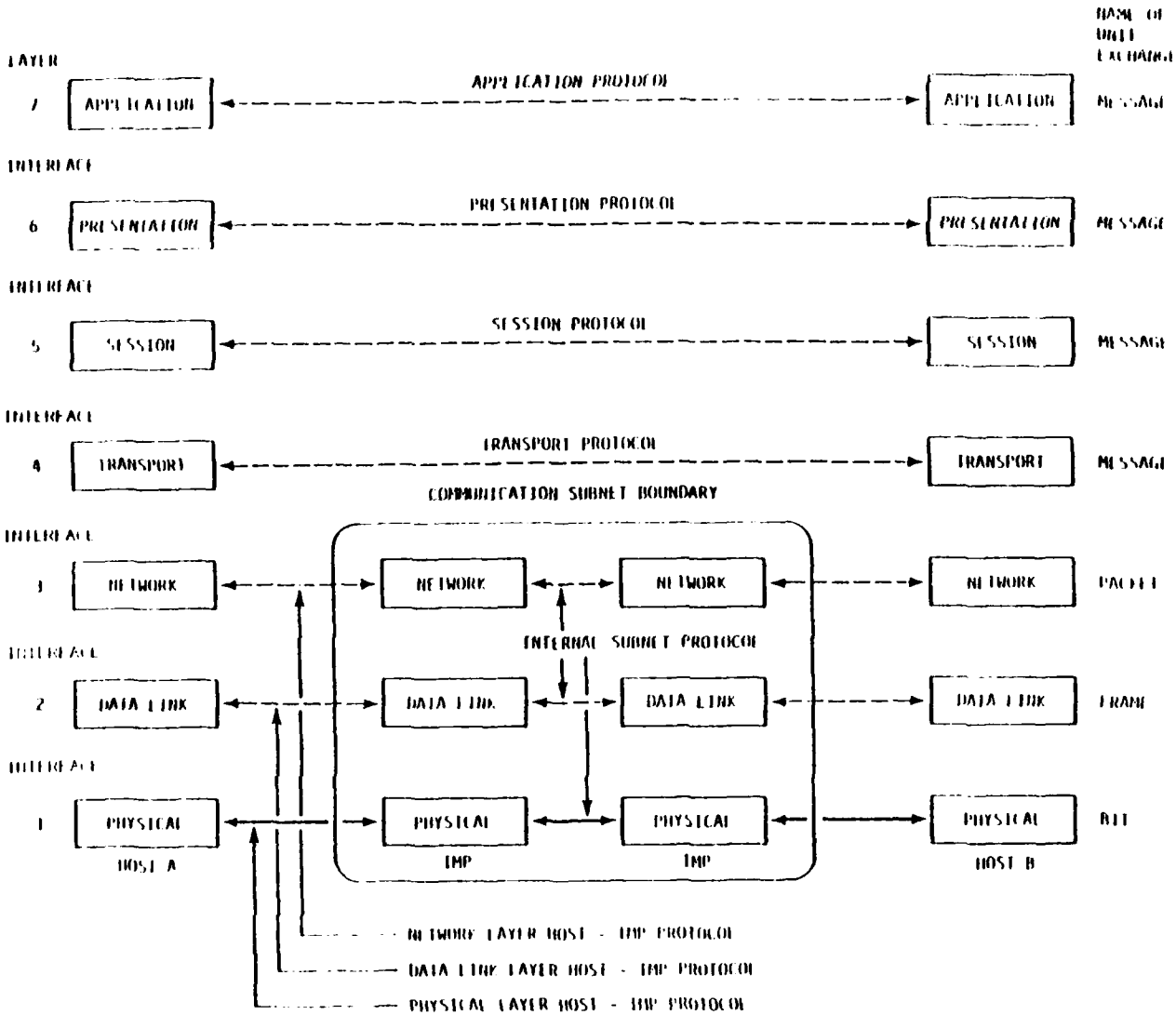



Figure 5.1-1 ISO Reference Model

designed network. Therefore, DSS should be capable of modeling such a design. We concluded, therefore, that a separate R/A simulator which modeled only reliability and availability would be a meaningless abstraction: The R/A Model should be embedded in a realistic model of a total DDP System so that the design choices inherent in the system could be evaluated as to their impact on reliability/availability. Further, the R/A modeling techniques that we have chosen should be transportable to any functioning simulator of a DDP system. The R/A Model has been designed to meet these two criteria (i.e., it must be embedded in a total system model and not be separate from it; and secondly, the techniques must be easily transportable to other simulators). We have chosen to embed the R/A Model in the communication protocol model since different protocols can have an enormous impact on the reliability/availability of the entire system. In fact, one of the major functions of a communication protocol is to correct and recover from transmission failures and nodal downtime. Therefore a R/A Model in this context would be particularly appropriate.

### 5.1.1 COMMUNICATION PROTOCOL (CP) DETAILED MODEL

Communication protocols are a relatively new area for simulation and modeling due to the advent of distributed processing technology. Communication protocols may be considered the operating system for networks. For DSS (Distributed System Simulator) to be useful in modeling networks it must be able to simulate communication at a detailed level. Network performance is significantly affected by the communication protocols that the network is using. User response time and throughput depend on the efficiency (degree of overhead) of the protocols. DSS, in order to be a practical tool, must facilitate the modeling of communication protocols. The most widely used communication protocol is the International Standards Organization's (ISO) High-Level Data Link Control (HDLC). The CP Model is a DSS detailed Model of HDLC and the X.25 interface which includes the following components:

- Number of Bits in an ACK Frame
- Number of Data Bits per Frame
- Interrupt the Processor Service Time
- Sender's Window Size
- Receiver's Window Size
- Negative Acknowledgement Frames
- Receive Ready Frames

### 5.1.2 RELIABILITY/AVAILABILITY (R/A) DETAILED MODEL

Increased reliability/availability is one of the most attractive features of distributed processing technology. Because a well designed network is a gracefully degradable system, traditional reliability models are not applicable. Reliability/availability is a function of a broad range of network characteristics, including hardware and system software components. Because of this, the R/A Detailed Model has been designed as a general model that can be used in a wide variety of environments - from distributed data base models (layer 7 of the ISO Reference Model) to data link control models (layer 2 of the ISO Reference Model). We have chosen to embed the R/A detailed model in the communication protocol model, though the routines and techniques used are just as applicable to any of the other detailed models. The R/A detailed model includes these components:

- System Availability
- System Reliability
- Failure Control
- Effects of Down Time
- Failure Recovery Rates
- Effects of Transmission Errors
- Effects of Lost Messages

The importance of the R/A detailed model stems from the fact that reliability and availability as performance issues cut across all boundaries and affect

all types of networks no matter what particular applications the networks are servicing. This is why the R/A detailed model has the broadest possible scope in terms of the general approach that it incorporates.

### 5.1.3 DISTRIBUTED DATABASE (DB) DETAILED MODEL

Distributed data bases are one of the most common application areas for distributed processing technology. There are several advantages to DDBM systems. As processor and memory costs are reduced, communication costs are becoming a larger percentage of the data processing budgets of many organizations. Local data bases, which can nevertheless be accessed from remote locations, reduce communication costs. Another advantage of DDBM systems is reliability. A particular site which is experiencing difficulties need not cause the entire network to fail.

A third advantage is that networks allow for incremental growth without restructuring the entire system. The performance characteristics of a DDBM system are dependent on many factors. Simulation models can help in the analyses that are necessary to design a cost-effective system that meets certain performance requirements. The DDBM System Model includes these components:

- Transaction Updates/Modifications
- Transaction Retrievals
- Number of Lockups
- Concurrency Control
- File Allocation
- Transaction Rates
- Transaction Overhead
- Processor Utilization
- Buffer Utilization
- Average Queue Lengths at File Servers
- Delays Due to Blocking/Synchronization
- Transaction Throughput
- Transaction Response Time

The functional characteristics of the model are such that the user may vary a wide range of DDBM system parameters and study the effects in terms of the resource utilizations (channels, processors, buffers, transmission paths) and the net user effect (throughputs, response times).

## 5.2 COMMUNICATION PROTOCOL MODEL

The Communication Protocol Model simulates the X.25 interface for packet switched service. Two new mechanisms - virtual circuits and datagrams are introduced in this model.

### 5.2.1 THE X.25 INTERFACE RECOMMENDATION

The X.25 Interface is the standard device independent interface between packet networks and user devices operating in packet mode. The X.25 Interface was recommended by the International Telegraph and Telephone Consultative Committee (CCITT) in 1976 to establish an international standard for public data networks. The X.25 interface recommendation at the packet level defines the following services:

1) Permanent virtual circuits (PVC);
2) Switched virtual circuits (SVC); and
3) Datagrams.

A permanent virtual circuit is a permanent association between two nodes, which is analogous to a point-to-point private line. Call setup or call clearing procedures are not required for a permanent virtual circuit. This type of virtual circuit was simulated in High Level Model 2 and is not included in the Communication Protocol model. A switched virtual circuit (SVC) creates a temporary association between an originating node and a target node. The node initiates a 'call request' to the network containing the address of the target node. A switched virtual circuit is established when the target node accepts the 'call request' and issues a 'call accepted' to the originator. Once the SVC has been established, the virtual circuit becomes a bidirectional, transparent, flow-controlled path between a pair of logical or

pnysical ports. Packets can now be sent from the originator to target. Once all packets have been received, a 'call clear' is sent from the target node to the originator node. The 'call clear', upon receipt at the originator, releases the temporary SVC.

The major benefit of SVC is the reduced overhead when exchanging large amounts of data packets periodically with a variety of target nodes. When short units of data need to be exchanged frequently, the overhead of SVC becomes significant compared to the amount of data being transferred. With small amounts of data the datagram can be more efficiently used. The datagram can be described as:

1) A self-contained message with a unique identifier, originator node and target node information,
2) Usually delivered intact, most often implemented as a single packet message,
3) Having a high probability of being delivered to the desired destination, however it can also be lost due to transmission errors or network problems,
4) The sequence of datagrams is not maintained,
5) Some form of error recovery for non-delivered datagrams is provided. [FOLT 80, RYBC 80].

This section of the User's Manual will focus on both SVC and datagrams as implemented in the Communication Protocol Model.

### 5.2.2 DATA FLOW DIAGRAMS

A Data Flow of a datagram traveling through a network is graphically portrayed in Figure 5.2.2-1. The more complex case of a Switched Virtual Circuit is displayed in Figure 5.2.2-2.

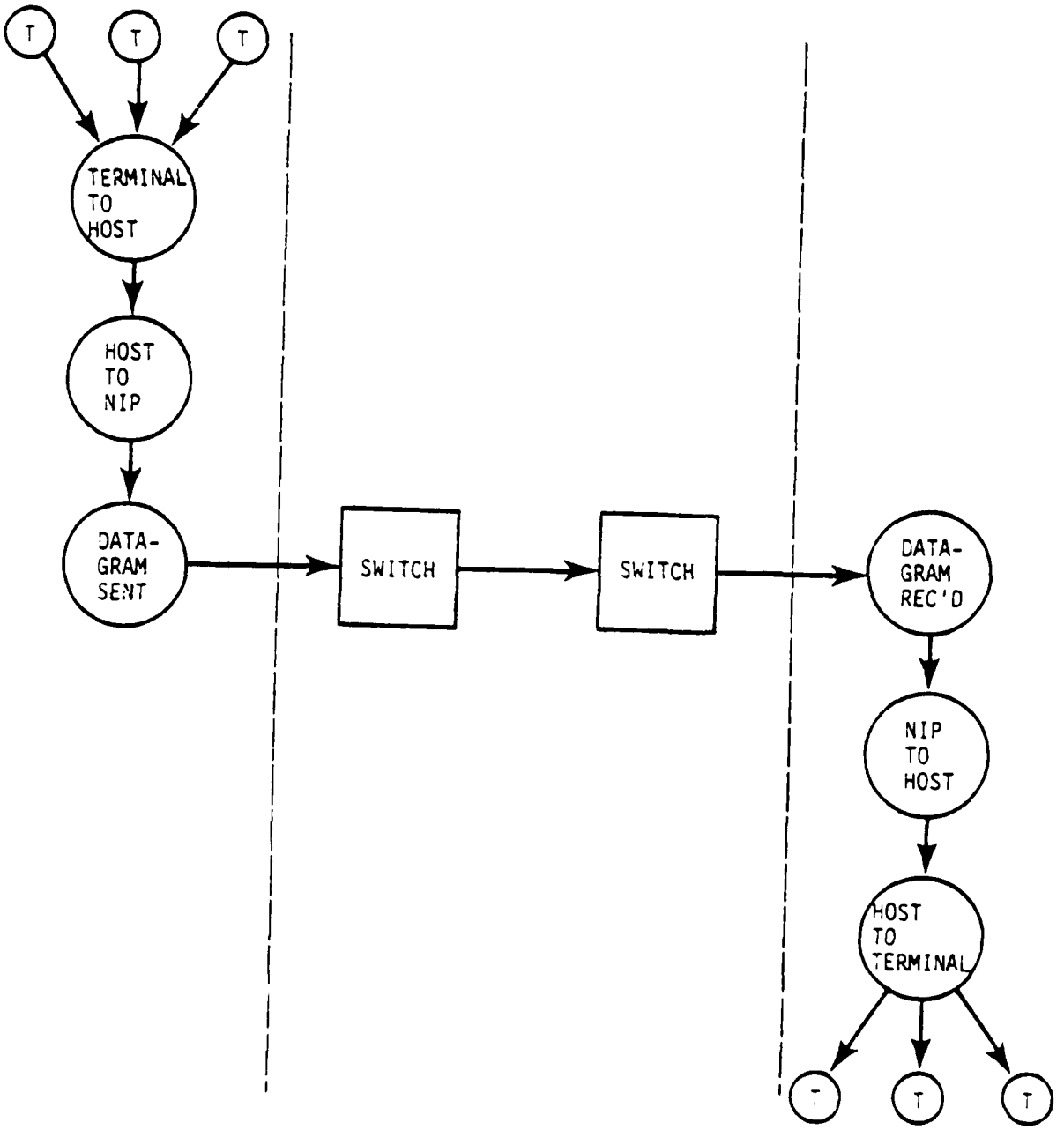
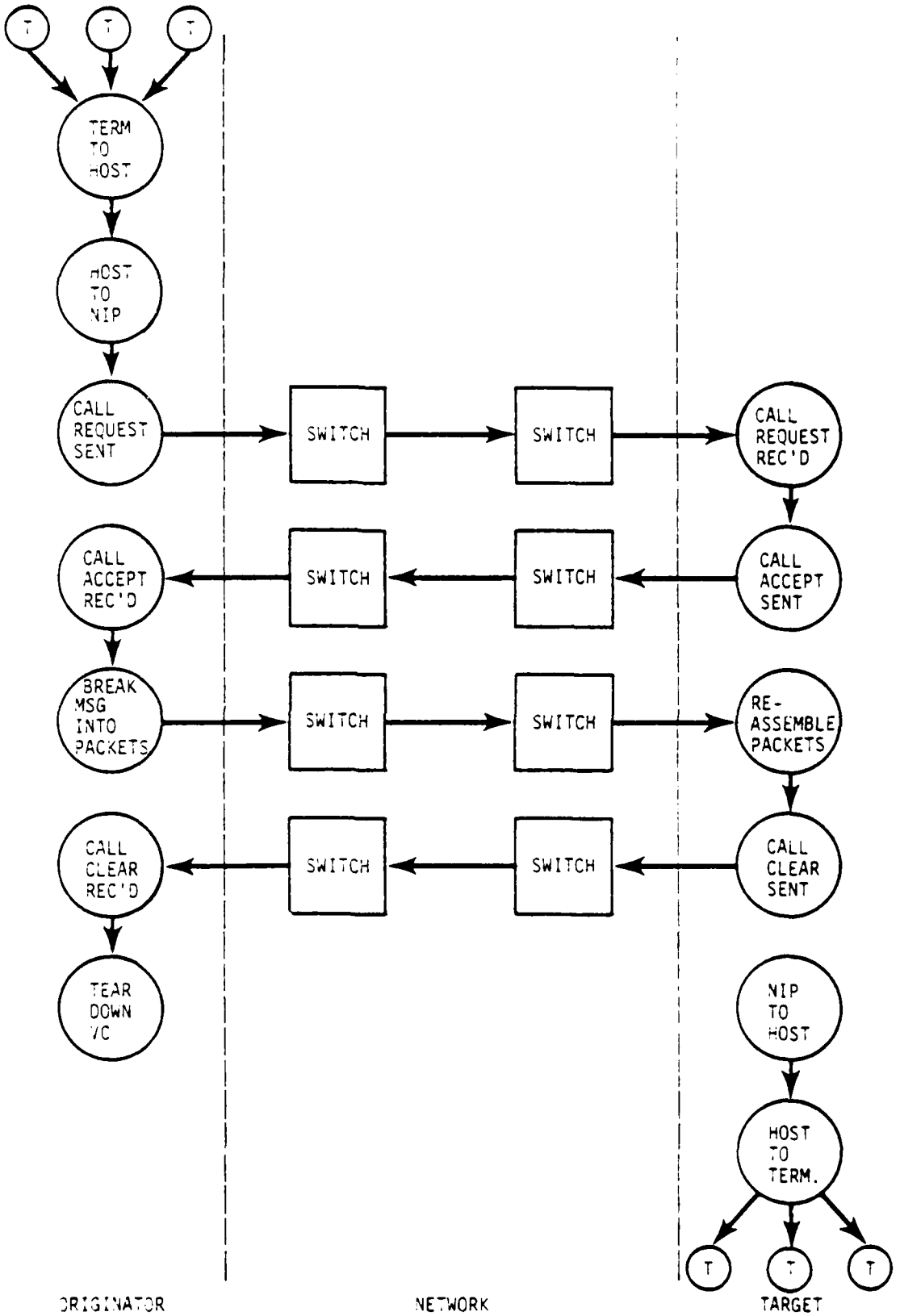



Figure 5.2.2-1 Datagram Data Flow

2A153/7A




2A153/6
00321

Figure 5.2.2-2 Switched Virtual Circuit Data Flow

### 5.2.3 MODEL ARCHITECTURE

The Communication Protocol Model consists of two model types:

- A model to describe the behavior of the host sites and
- A model to describe the benavior of switching nodes.

#### 5.2.3.1 Host Sites

The function of the host site is to generate new messages and destroy messages sent from other host nodes. Each host site consists of:

- A Host processor
- A Network Interface Processor (NIP)
- A variable number of internodel transmission devices
- 2 Channels
- 1 Buffer

The host sites in the Detail Models differs from host sites in HLM Models (Section 3) by allowing host to perform both host functions and switching functions. Thus, one model (Model 1 - Host Sites) would have been sufficient to simulate the different complexity levels, however two models - Host and Switch - were retained. The Switching Model is available for simulating nodes performing switching functions exclusively.

Figure 5.2.3.1-1 graphically portrays the Host Site architecture.

Table 5.2.3.1-1 provides a summary of the System Description for the Host Sites.

Table 5.2.3.1-1 System Description for Host Site

| DEVICE NAME | ECSS DEVICE TYPE | PERFORMANCE SPECIFICATIONS |
|---|---|---|
| D01A.CPU (Processor) | EXECUTIONS AND JOB STORE DEVICE | 500,000 INSTRUCTIONS /SEC |
| D01A.NIP (Network Interface Processor) | NETWORK INPUT PROCESSOR EXECUTES AND JOB STORE DEVICES | 500,000 INSTRUCTIONS /SEC |
| T01A.M01 (Internodal Transmission Device) | TRANSMISSION DEVICE | 20,000 BYTES/SEC |
| D01B. CHANNELS (Channel) | TRANSMISSION DEVICE | 20,000 BYTES/SEC |
| D01D.BUF (Buffer) | STORAGE DEVICE | 10,000 BYTES |
| D01C.TERMINAL (Terminal) | I/O DEVICE | 9600 BYTES/SEC |
| P01A.T.PATH (Internodal Path) | PATH | CONNECT D01B. CHANNELS TO D01C TERMINALS |

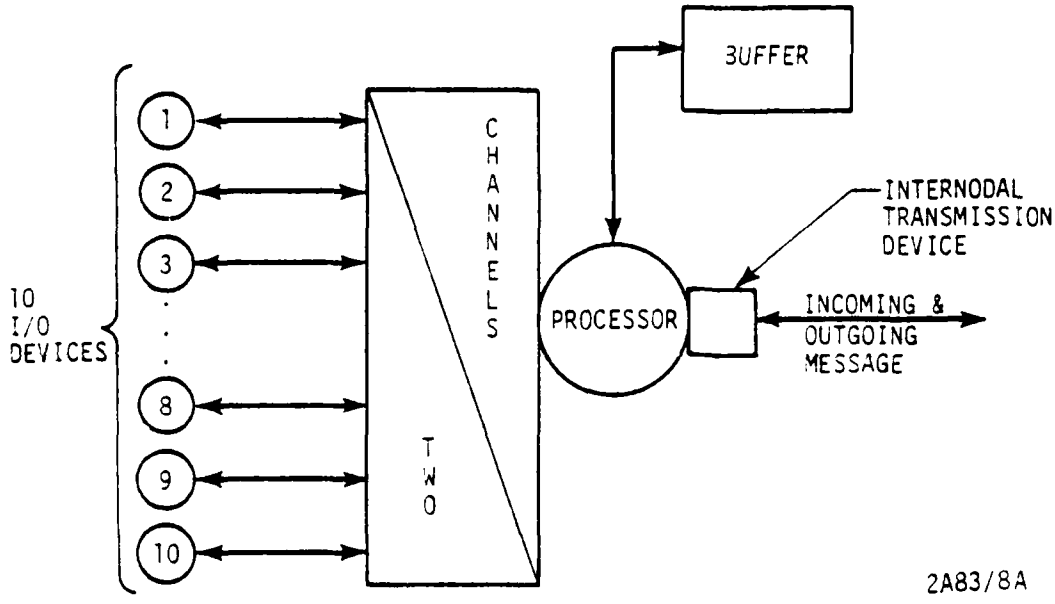



Figure 5.2.3.1-1 Host Site Configuration

5.2.3.2 Switching Nodes

The function of the switching node is to accept and forward messages along the network. Each switching node consists of:

- A Processor
- A variable number of Internodal Transmission devices
- A Buffer

The Processor and Buffer declarations are provided in the DSS Switching Model. The specifications for the Internodal Transmission devices are defined in the TP.FILE. Figure 5.2.3.2-1 graphically displays this configuration. Table 5.2.3.2-1 provides a summary for the System Description Section for the switching node.

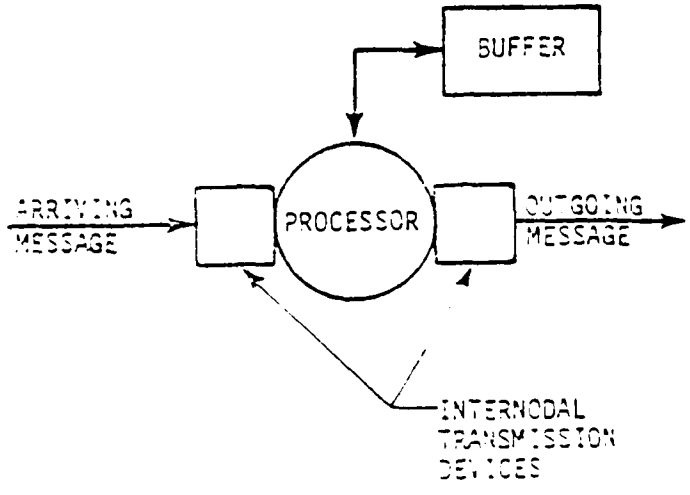



Figure 5.2.3.2-1 Switching Node Configuration (Dedicated Option)

Table 5.2.3.2-1 System Description for Switching Node

| DEVICE NAME | DEVICE TYPE | ECSS PERFORMANCE SPECIFICATIONS |
|---|---|---|
| D02A.CPU (Processor) | EXECUTION AND JOB STORE DEVICE | 500,000 INSTRUCTIONS/SEC |
| T02A.M01 (Internodal Transmission Device) | TRANSMISSION DEVICES | 30,000 BYTES/SEC |
| D02B.BUF (Buffer) | STORAGE DEVICE | 10,000 BYTES |
| P01A.N01.N02 (Internodal Path) | PATH | CONNECTS T01A.M01 TO T02A.M02 |

5.2.4 FUNCTIONAL LOGIC FLOW

The Communication Protocol Model simulates a Switched Virtual Circuit (SVC) Packet Service Network that includes Datagrams. Host sites contain a Host Processor which generates messages and forwards them to the host site Network Interface Processor (NIP). If the message is a Datagram, then the NIP will interject the message into the network and it will travel to its destination in a manner similar to the messages in High Level Models 1 and 2.

If the message requires a SVC, the virtual circuit will be allocated as a connecting log of nodes from originator to target. For example,

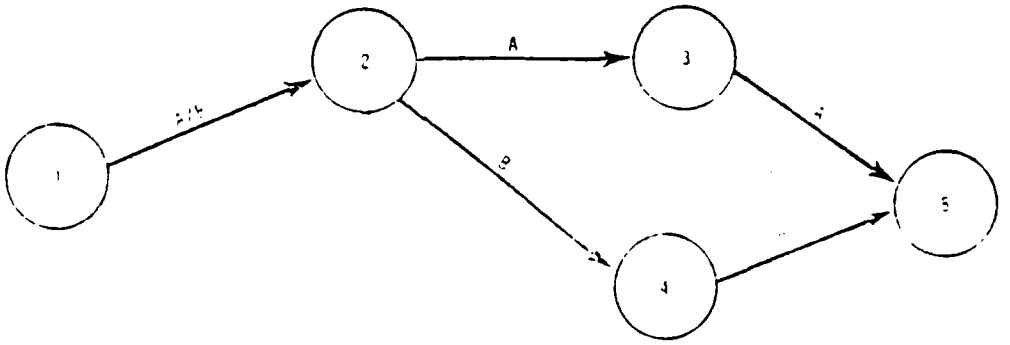



Figure 5.2.4-1 CP Topology - Level C

given the 5 node network illustrated in Figure 5.2.4-1, with node 1 as the originator and node 5 as the target, two possible virtual circuits exist:

- VC A - 1 to 2 to 3 to 5
- VC B - 1 to 2 to 4 to 5

Whichever VC is taken by both the 'Call Request' and 'Call Accept' messages, that same VC will be used for all data packets of the original messages. Switching sites accept and forward messages. The next node to forward a message is determined by the allocated VC. Finally, the target NIP accepts the packets, reassembles the packets into the original message, updates message statistics, passes the message to its Host Processor and sends a 'call clear' back to the originator.

Figure 5.2.4-2 is a High Level Functional Flow of this process. Table 5.2.4-1 lists the message types while Table 5.2.4-2 provides a detailed functional flow of the Communications Protocol Model. The message types 4 through 9 are High Level Data Link Control (HDLC) message types. HDLC is one of the most popular "bit-oriented" Data Link Control (DLC) protocols. The purpose of a DLC Protocol is to assure the bit stream received is an error-free replica of the bit stream transmitted [GREE 80].

Table 5.2.4-1 Communication Protocol Message Types

| MESSAGE TYPE | DESCRIPTION |
|---|---|
| 1 | TERMINAL TO HOST |
| 2 | HOST TO NIP |
| 3 | RECEIVE AT NIP - CHECK IF VIRTUAL CIRCUIT REQUIRED |
| *4 | CALL REQUEST |
| *5 | CALL ACCEPTED |
| *6 | DATA PACKETS |
| *7 | ACKNOWLEDGEMENT BETWEEN NODES |
| *8 | PACKETS REASSEMBLED INTO ORIGINAL MESSAGE, PASSED TO HOST APPLICATION LEVEL |
| *9 | CLEAR REQUEST |
| 10 | ORIGINAL MESSAGE TO BE RETRANSMITTED |

* HDLC MESSAGE TYPES

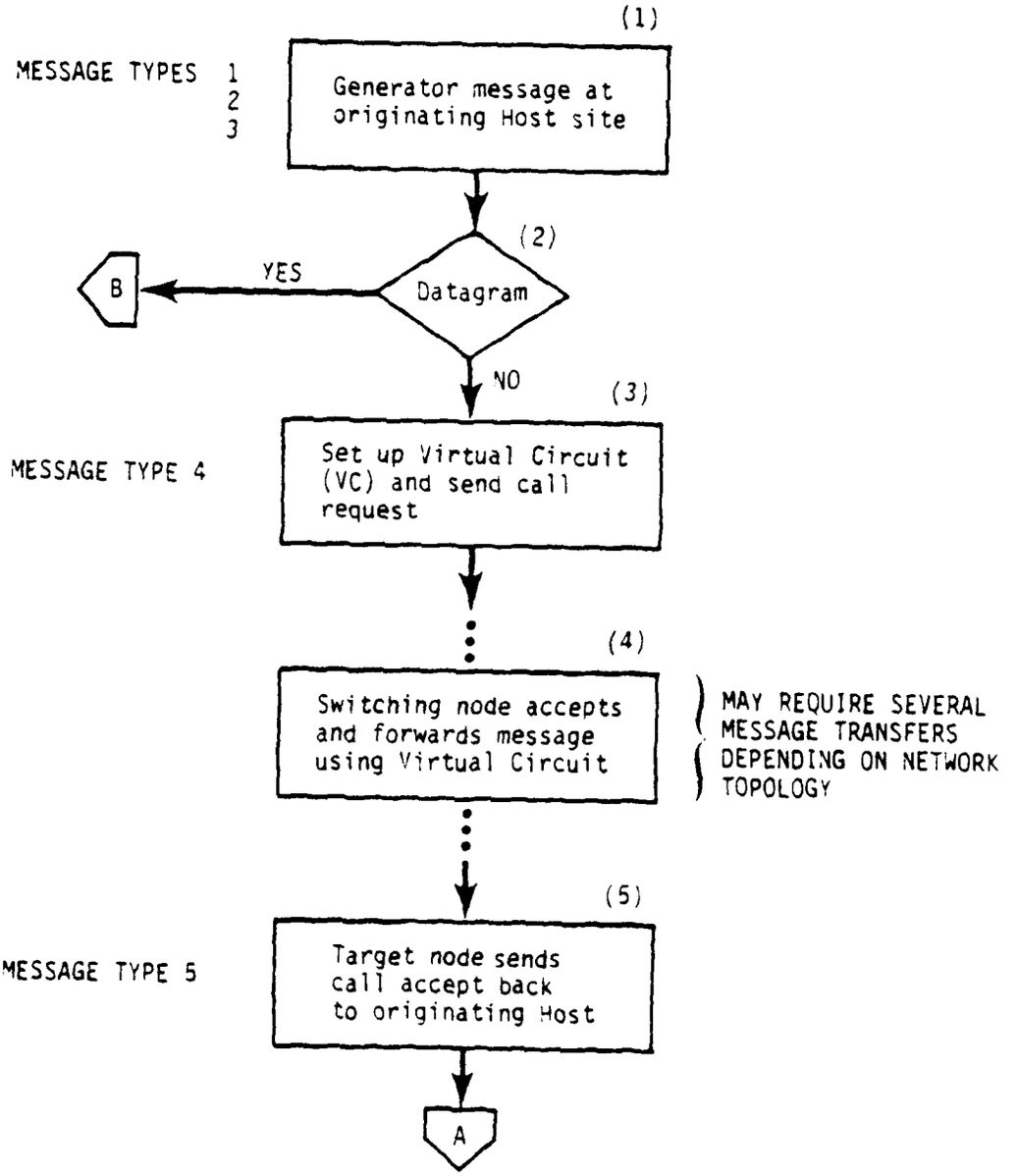
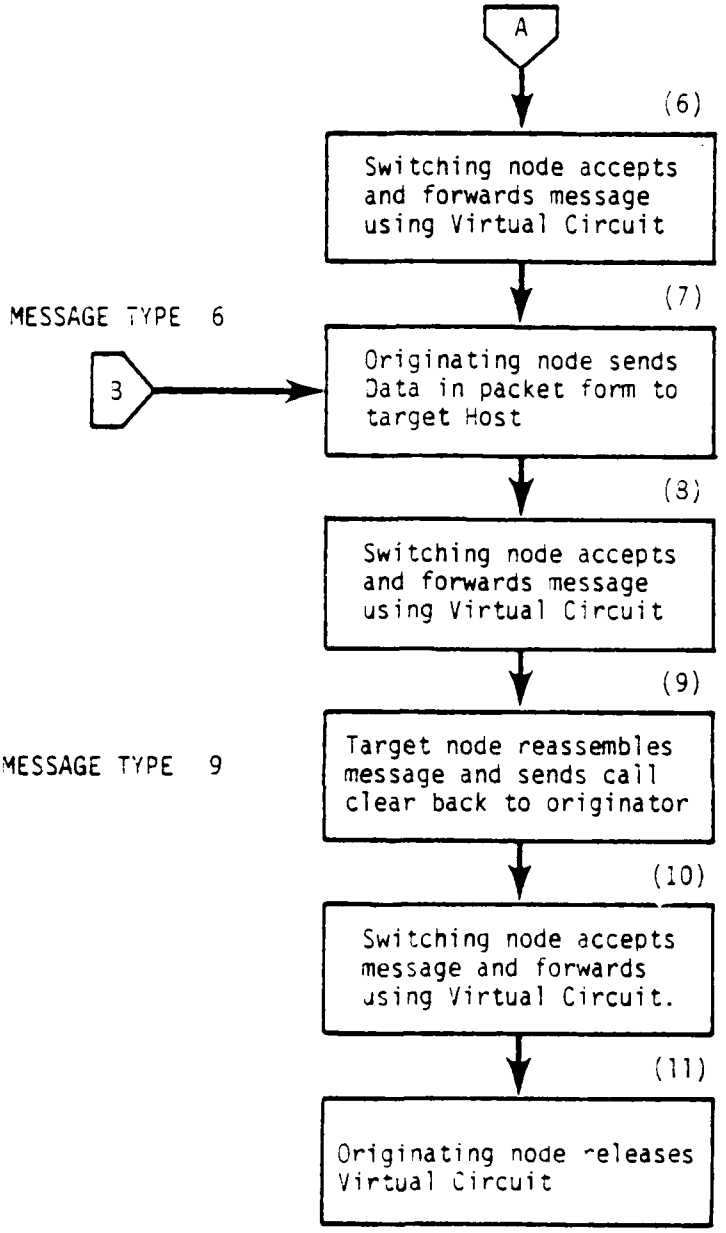



Figure 5.2.4-2 High Level Functional Flow of Communication Protocol Model




2A153/4

Figure 5.2.4-2 High Level Functional Flow of Communication Protocol Model (CONTINUED)

Table 5.2.4-2 Detailed Functional Flow Communication Protocol Model

| MESSAGE TYPE | AT ORIGINATION NODE | IN TRANSIT | AT TARGET NODE | MODEL MODULE<br>P-PROCESS<br>R-ROUTINE<br>J-JOB |
|---|---|---|---|---|
| 1 | • GENERATE MESSAGE AT TERMINALS<br>• SEND FROM TERMINALS HOST | | | HO. MSG. AR (R)<br>AAR (P)<br>CP (P)<br>TH (J) |
| 2 | • SEND MESSAGE FROM HOST TO NIP | | | CP (P)<br>HN (J) |
| 3 | • IF A DATAGRAM CREATE A MESSAGE TYPE 6 ELSE<br>• SET UP VIRTUAL CIRCUIT | | | CP (P)<br>VC. REQUEST (R) |
| 4 | • SEND A CALL REQUEST TO TARGET NODE | | | CP (P)<br>SM (J)<br>VC. NODE (R) |
| 4 | | • ACKNOWLEDGE PRIOR NODE TYPE 7 MESSAGE<br>• FORWARD TO NEXT NODE ON VIRTUAL CIRCUIT | | CP (P)<br>AK (J)<br>SM (J)<br>VC. NODE (R) |
| | | | • ACKNOWLEDGE MESSAGE<br>• INVERT VIRTUAL CIRCUIT | CP (P)<br>AK (J)<br>V.C. INVERT (R) |

Table 5.2.4-2 Detailed Functional Flow for Communication Protocol Model
(Continued)

| MESSAGE TYPE | AT ORIGINATION NODE | IN TRANSMIT | AT TARGET NODE | MODULE<br>P-PROCESS<br>R-ROUTINE<br>J-JOB |
|---|---|---|---|---|
| 5 | | | • SEND CALL ACCEPT TO ORIGINATING NODE | CP (P)<br>SM (J)<br>VC. NODE (R) |
| 5 | | • ACKNOWLEDGE MESSAGE - TYPE 7<br>• FORWARD MESSAGE | | CP (P)<br>AK (J)<br>SM (J)<br>VC. NODE (R) |
| 5 | • ACKNOWLEDGE MESSAGE MESSAGE - TYPE 7<br>• INVERT VC<br>• BREAK MESSAGE INTO PACKETS | | | CP (P)<br>AK (J)<br>VC INVERT (R)<br>PACKET GEN (J) |
| 6 | • FORWARD PACKETS | | | PC (P)<br>SM (J) |
| | • ACKNOWLEDGE PACKETS - TYPE 7<br>• FORWARD PACKETS | | | PC (P)<br>AK (J)<br>SM (J)<br>VC.NODE (R) |
| 6 | | • FORWARD PACKETS | | PC (P)<br>SM (J) |
| | | • ACKNOWLEDGE PACKETS - TYPE 7<br>• FORWARD PACKETS | | PC (P)<br>AK (J)<br>SM (J)<br>VC.NODE (R) |
| 6 | | | • ACKNOWLEDGE PACKET - TYPE 7<br>• REASSEMBLE PACKETS INTO ORIGINAL MESSAGE<br>• PASS MESSAGE TO APPLICATION LEVEL | PC (P)<br>AK (J)<br>RA (J) |

Table 5.2.4-2 Detailed Functional Flow for Communication Protocol Model
(Continued)

| MESSAGE TYPE | AT ORIGINATION NODE | IN TRANSIT NODE | AT TARGET NODE | MODULE<br>P-PROCESS<br>R-ROUTINE<br>J-JOB |
|---|---|---|---|---|
| 8 | | | • SEND REASSEMBLED MESSAGE FROM NIP TO HOST<br>• SEND FROM HOST TO TERMINALS | CP (P)<br>NH (J)<br>HT (J) |
| 9 | | | • SEND CALL CLEAR | CP (P)<br>CM (J)<br>VC. NODE (R) |
| 9 | | • ACKNOWLEDGE MESSAGE (TYPE 7)<br>• FORWARD MESSAGE | | CP (P)<br>AK (J)<br>SM (J)<br>VC. NODE (R) |
| 9 | • ACKNOWLEDGE MESSAGE (TYPE 7)<br>• RELEASE VC | | | CP (P)<br>AK (J) |

Table 5.2.4-2 Detailed Functional Flow for Communication protocol Model
(Continued)

| MESSAGE TYPE | AT ORIGINATION NODE / IN TRANSIT NODE / AT TARGET NODE | MODULE P-PROCESS R-ROUTINE J-JOB |
|---|---|---|
| 7 | • DELETE ORIGINAL MESSAGE FROM UNACKNOWLEDGE LIST<br>• PREVENT RETRANSMISSION OF BACKUP MESSAGE<br>• THIS ACKNOWLEDGEMENT MESSAGE IS REPORTED FOR ALL COMPLETED NODE TO NODE TRANSMISSIONS EXCEPT DATAGRAM TRANSMITTAL. | CP (P) |
| 10 | • THIS IS THE BACKUP COPY OF A MESSAGE SENT. IT IS RETRANSMITTED IF AN ACKNOWLEDGEMENT IS NOT RECEIVED WITHIN THE TIME OUT PERIOD | CP (P)<br>PC (J)<br>..DM (J)<br>RM (J)<br>SM (J)<br>VC. NODE (R) |

Two major external processes provide the control functions (e.g., message generation and starting computer jobs) for the communication protocol model. They are:

- ARR process (message generation)
- CP Process (communication process - controlling computer jobs)

A brief description of each process is provided:

Message Generation - External Process ..ARR

The ..ARR process provides the vehicle for generating every message introduced into the network. This process simply cycles according to the interarrival rate (MIT) established in the MESS file or defaulted in the EXEC file. Messages are created each time the process cycles. The number of instances of ..ARR in the external Process Reports indicates the number of different message types generated during the simulation run.

Communication Process - External Process CP

The primary function of the communication process is to control the communication between nodes. A communication process may start jobs on a processing unit which simulates the activity of transferring, switching and receiving messages. Statistics on these external processes are displayed on the External Process Report. The number of instances for the CP processes should be one. These processes are started only once and continue to operate for the full duration of the simulation.

The communication process activates and controls these jobs:

- JOB SM
- JOB RM
- JOB HT
- JOB TH
- JOB HN
- JOB NH
- JOB AK
- JOB PA
- JOB PC

A description of these jobs is provided.

JOB SM - Message Transferring

JOB SM simulates the activity of transfering a message from a current site to the next node. It accomplishes this task by filing the pointer to the message (MSG) data in the switching nodes message file (..MGFILE). The JOB concludes by signaling the switching node. This notifies the communication process that it has a message waiting for it. The total number of instances for a unique JOB SM can be interpreted as the total number of messages transmitted from one site to another.

JOB RM - Retransmit Message

Job RM simulates the retransmission of a message if an acknowledgement has not been received within a user specified time. The number of instances of JOB RM indicates the number of packets sent from a particular node which were rejected by an adjacent node.

JOB AK - Acknowledgement

Job AK is responsible for sending acknowledgements for accepted packets. This job is only operational when a "1" is indicated for "ACK" in the EXEC file. The number of instances of JOB AK indicates the number of packets accepted by a particular node.

JOB PA - Packet Retransmit

Job PA simulates the reassembly of a message at the NIP. The number of instances of Job PA indicates the total number of completely reassembled messages at a host node.

JOB PC - Post Communication

Job PC is the timing mechanism that schedules the retransmit command for all unacknowledged packets. The user specifies the frequency with which this job is performed by selecting the maximum response time tolerable for all acknowledgements at the host (TMH) and switching nodes (TMC). The number of instances of this job indicates how many times a particular node assessed the status of its outgoing packets.

JOB TH- Terminal to Host Transmission

This job simulates the transferring of a message from a host site's terminal to the host processor. Tasks accomplished by this job are:

- Sending appropriate bytes from a randomly selected terminal via a randomly selected channel,
- Message type is changed from a new message (Type 1) to 'received at Host', (Type 2),
- Message is then filed in current node's message file,
- CP is signaled.

Number of unique occurances of this job can be interpreted as total messages received at that Host from its users.

JOB HN- Host to NIP Transfer

This job increases simulation time to model the overhead incurred by the host CPU when relaying a message to its Network Interface Processor (NIP). JOB HN accomplisnes this by changing message type to 'received at NIP' (Type 3), filing the message in its message file and signaling CP. Total number of instances for this job for any given node, represents total number of messages entered into network.

JOB NH- NIP to Host Transfer

This job incurs simulation time to represent overhead when messages flow from a Host site NIP to its Host CPU. Total number of instances for that node represent number of messages received by that node from the network.

JOB HT - Host to Terminal Transfer

This job simulates the transferring of a message from a Host CPU to its terminals. It accomplishes this by sending appropriate bytes via a randomly selected channel to a randomly selected terminal. Total number of instances for this job for any given node represents communication activity between CPU and its terminals. Refer to DSS User's Manual for more information relating to Job/External Process reports.

### 5.2.5 SIMULATION OUTPUT

The DSS Simulator produces two output reports:

- Message Statistics
- ECSS Standard Reports
  (See Appendix D for examples.)

For a description of each report contained in this section refer to the User's Manual. For a complete sample of these reports refer to Appendix D

## 5.3 RELIABILITY/AVAILABILITY MODEL

The Reliability/Availability Model extends the Communication Protocol Model by accounting for nodal failures and network transmission errors. Two new mechanisms - rejections and adaptive routing are introduced in this model.

### 5.3.1 NETWORK RELIABILITY

Reliability is a major concern in communications oriented systems. Communication lines can fail to give adequate service in one of the following ways:

- Line errors - Some data transmitted over the line are garbled due to bits being added or lost.
- Line Outages - The connecting line can become unavailable causing transmissions to be blocked if an alternative line cannot be provided.
- Engaged or Busy Signals - A target node may be at maximum capacity in respect to network inputs or outputs and thus unavailable.
- Disconnections - A node on the network can 'crash' and disconnect itself from the network until it recovers. If the node is a host, internal processing may continue while isolated from the network.

Certain steps can be taken to improve reliability. They include:

- Automatic Retry - When the line fails to send the data and no acknowledgement is received, the sender will retransmit the message. This process is of most value when outages are of short durations.

- Alternate Paths - The network communication lines can be duplicated in part, for reliability. Two or more communication lines may share the load from a given location to another. When one line fails, service is available on the alternate line. In a packet - switching network, packets can traverse the network, by passing parts of the network that have failed. Packet switching thus forms the basis of an adaptive network that automatically adjusts the route which the packet traverses, to bypass failed lines or failed nodes.
- Buffering - As a 'last resort' tactic, messages blocked at a sending node can be buffered until the required circuit becomes available.

The Reliability/Availability Model uses all the above methods to compensate for line and nodal failures.

5.3.2 LAYERED APPROACH TO BUILDING RELIABILITY/AVAILABILITY MODEL

The Communication Protocol Model described in 4.2 formed the foundation for the new Reliability/Availability Model. The communication mechanisms developed in the Communication Protocol Model were only slightly adapted to accomodate the network support system required by the Reliability/Availability Model. New jobs were added with minimal impact to existing jobs.

Upward compatability of the CP model was retained, since the CP model could be replicated by the R/A model if the user specifies no transmission error and no nodal crashes.

5.3.3 MODEL ARCHITECTURE

The R/A model consists of two model types:

- A model to describe the behavior of the host sites and
- A model to describe the behavior of switching nodes.

5.3.3.1 Host Sites

The function of the host site is to generate new messages and destroy messages sent from other host nodes. Each host site consists of:

- A Host processor
- A Network Interface Processor (NIP)
- A variable number of internodal transmission devices
- 2 Channels
- 1 Buffer

The host sites in the Detail Models differs from host sites in HLM Models (Section 4) by allowing host to perform both host functions and switching functions. Thus, one model (Model 1 - Host Sites) would have been sufficient to simulate the different complexity levels; however two models - Host and Switch - were retained. The Switching Model is available for simulating nodes performing switching functions exclusively.

Figure 5.3.3.1-1 graphically portrays the Host Site architecture.

Table 5.3.3.1-1 provides a summary of the System Description for the Host Sites.

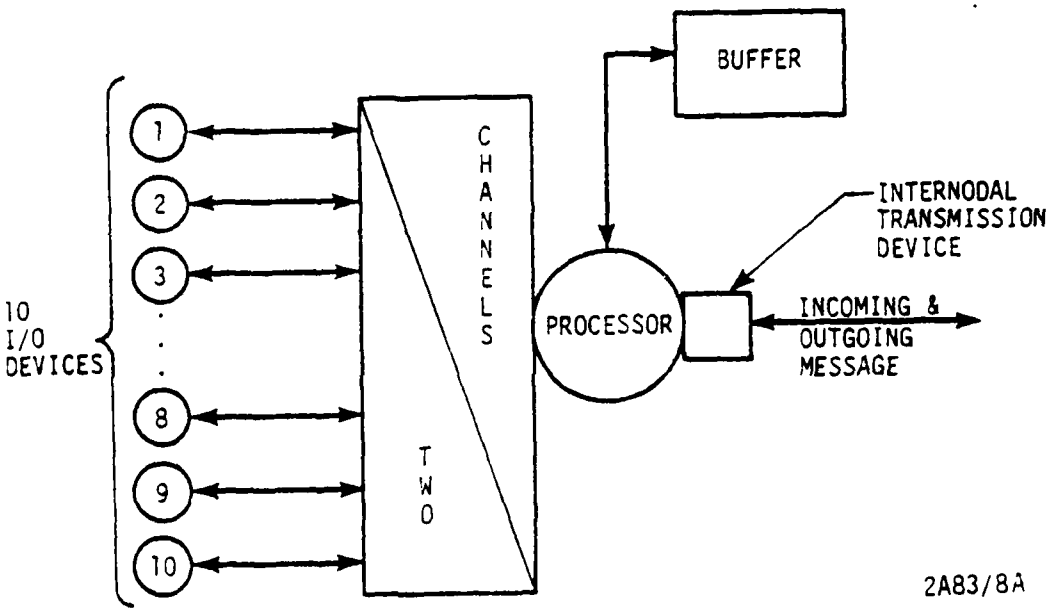



Figure 5.3.3.1-1 Host Site Configuration

Table 5.3.3.1-1 Host Site Configuration

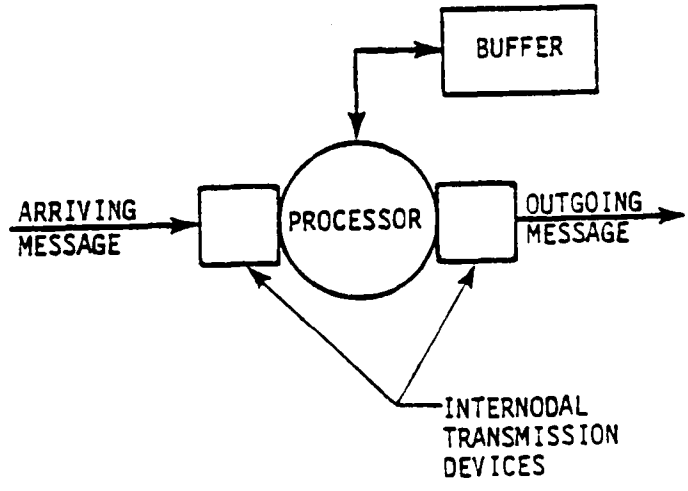
| DEVICE NAME | ECSS DEVICE TYPE | PERFORMANCE SPECIFICATIONS |
|---|---|---|
| D01A.CPU (Processor) | EXECUTIONS ANDS JOB STORE DEVICE | 500,000 INSTRUCTIONS /SEC |
| D01A.NIP (Network Interface Processor) | NETWORK INPUT PROCESSOR EXECUTES AND JOB STORE DEVICES | 500,000 INSTRUCTIONS /SEC |
| T01A.M01 (Internodal Transmission Device) | TRANSMISSION DEVICE | 20,000 BYTES/SEC |
| D01B. CHANNELS (Channel) | TRANSMISSION DEVICE | 20,000 BYTES/SEC |
| D01D.BUF (Buffer) | STORAGE DEVICE | 10,000 BYTES |
| D01C.TERMINAL (Terminal) | I/O DEVICE | 9600 BYTES/SEC |
| P01A.T.PATH (Internodal Path) | PATH | CONNECT D01B. CHANNELS TO D01C TERMINALS |

5.3.3.2 Switching Nodes

The function of the switching node is to accept and forward messages along the network. Each switching node consists of:

- A Processor
- A variable number of Internodal Transmission devices
- A Buffer

The Processor and Buffer declarations are provided in the DSS Switching Model. The specifications for the Internodal Transmission devices are defined in the TP.FILE. Figure 5.3.3.2-1 graphically displays the switching node configuration. Table 5.3.3.2-1 provides a summary for the System Description Section for the switching node.




2A83/6

Figure 5.3.3.2-1 Switching Node Configuration (Dedicated Option)

Table 5.3.3.2-1 System Description for Switching Node

| DEVICE NAME | DEVICE TYPE | ECSS PERFORMANCE SPECIFICATIONS |
|---|---|---|
| D02A.CPU (Processor) | EXECUTION AND JOB STORE DEVICE | 500,000 INSTRUCTIONS/SEC |
| T02A.M01 (Internodal Transmission Device) | TRANSMISSION DEVICES | 30,000 BYTES/SEC |
| D02B.BUF (Buffer) | STORAGE DEVICE | 10,000 BYTES |
| P01A.N01.N02 (Internodal Path) | PATH | CONNECTS T01A.M01 TO T02A.M02 |

5.3.4 FUNCTIONAL LOGIC FLOW

Two major external processes provide the control functions (e.g., message generation and starting computer jobs) for the R/A model. They are:

- ..ARR process (message generation)
- CP Process (communication process - controlling computer jobs)

A brief description of each process is provided:

Message Generation - External Process ..ARR

The ..ARR process provides the vehicle for generating every message introduced into the network. This process simply cycles according to the interarrival rate (MIT) established in the MESS file or defaulted in the EXEC file. Messages are created each time the process cycles. The number of instances of ..ARR in the external Process Reports indicates the number of different message types generated during the simulation run.

Communication Process - External Process CP

The primary function of the communication process is to control the communication between nodes. A communication process may start jobs on a processing unit which simulates the activity of transferring, switching and receiving messages. Statistics on these external processes are displayed on the External Process Report. The number of instances for the CP processes should be one. These processes are started only once and continue to operate for the full duration of the simulation.

The communication process activates and controls these jobs:

- JOB SM
- JOB RM
- JOB HT
- JOB TH
- JOB HN
- JOB NH
- JOB AK
- JOB PA
- JOB PC
- JOB RC
- JOB RD

A description of these jobs is provided.

JOB SM - Message Transferring

JOB SM simulates the activity of transfering a message from a current site to the next node. It accomplishes this task by filing the pointer to the message (MSG) data in the switching nodes message file (..MGFILE). The JOB concludes by signaling the switching node. This notifies the communication process that it has a message waiting for it. The total number of instances for a unique JOB SM can be interpreted as the total number of messages transmitted from one site to another.

JOB RM - Retransmit Message

JOB RM simulates the retransmission of a message if an acknowledgement has not been received within a user specified time. The number of instances of Job RM indicates the number of packets sent from a particular node which were rejected by an adjacent node.

JOB AK - Acknowledgement

JOB AK is responsible for sending acknowledgements for accepted packets. This job is only operational when a "1" is indicated for "ACK" in the EXEC file. The number of instances of JOB AK indicates the number of packets accepted by a particular node.

JOB PA - Packet Assembly

JOB PA simulates the reassembly of a message at the NIP. The number of instances of JOB PA indicates the total number of completely reassembled messages at a host node.

JOB PC - Post Communication

JOB PC is the timing mechanism that schedules the retransmit command for all unacknowledged packets. The user specifies the frequency in which this job is performed by selecting the maximum response time tolerable for all acknowledgements at the host (TMH) and switching nodes (TMC). The number of instances of this job indicates how many times a particular node assessed the status of its outgoing packets.

JOB TH- Terminal to Host Transmission

This job simulates the transfering of a message from a host site's terminal to the host processor. Tasks accomplished by this job are:

- Sending appropriate bytes from a randomly selected terminal via a randomly selected channel,
- Message type is changed from a new message (Type 1) to 'received at Host', (Type 2),
- Message is then filed in current node's message file,
- CP is signaled.

Number of unique occurances of this job can be interpreted as total messages received at that host from its users.

JOB HN- Host to NIP Transfer

This job increases simulation time to model the overhead incurred by the HOST CPU when relaying a message to its Network Interface Processor (NIP). JOB HN accomplishes this by changing message type to received at NIP' (Type 3), filing the message in its message file and signaling CP. Total number of instances for this job for any given node represents total number of messages entered into network.

JOB NH- NIP to Host Transfer

This job incurrs simulation time to represent overhead when messages flow from a Host site NIP to its Host CPU. Total number of instances for that node represent number of messages received by that node from the network.

JOB HT - Host to Terminal Transfer

This job simulates the transferring of a message from a Host CPU to its terminals. It accomplishes this by sending appropriate bytes via a randomly selected channel to a randomly selected terminal. Total number of instances for this job for any given node represents communication activity between CPU and its terminals.

JOB RC - Reroute for Congestion

This job simulates the selection of a new virtual circuit when the next node in the current virtual circuit is out of buffer space. Total number of instances represents network rerouting due to traffic overload.

JOB RD - Reroute because of Nodal Failure

This job simulates the selection of a new virtual circuit when the next node in the current virtual circuit is disconnected from the network. Total number of instances represents network rerouting due to nodal failures.

Two new external processes were added to the R/A model to simulate the failure of nodal sites. They are:

EXTERNAL PROCESS - RECOV

Controls the time interval a node is unavailable to the network. When the External Process RECOV is signalled after the duration of the crash, the node's status is toggled to 1 (up).

EXTERNAL PROCESS - CRASH

Controls the time interval between failures for a network node. After a time interval has elapsed, the External Process CRASH is signaled and the node's status is toggled to Ø (Down).

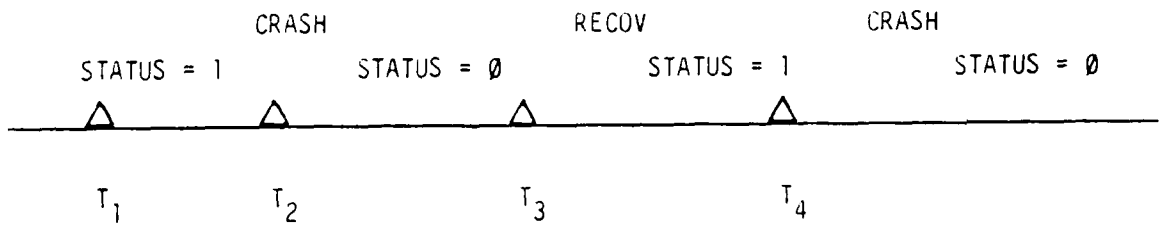



Figure 5.3.4-1 External Process CRASH and RECOV Relationship

Figure 5.3.4-1 shows the relationship between R/A model's external process CRASH and external process RECOV (see 5.3.4.2 Nodal Failures).

From a functional perspective, three new mechanism were added to the C/P model to implement the R/A model. The three mechanisms were:

1 Transmission Errors
2 Nodal Failures
3 Adaptive Routing

The following sections will describe each mechanism in detail.

5.3.4.1 Transmission Errors

Transmission errors are modeled in the Reliability/Availability Model by inputting a user specified reject factor. The user can specify a reject factor of less than 1.0 (e.g., .20 for 20%); its complement would of course, be the site's reliability factor. If no reject factor (REJ) is specified in the EXEC file, the model will default REJ to 0.

The functional logic flow for rejects is displayed in Figure 5.3.4.1-1. If a message arrives at a node other then the originator, and it is not an acknowledgement (type = 7) and not an intranodal message (type = 1, 2, 3 or 8), then the message is tested for rejection. The rejection test involves a uniform random number in the open interval (0,1) being compared to the reject factor. When a message is rejected by a receiving node, no acknowledgement will be sent. Once the maximum time out (TMO) has been exceeded, the sending node will retransmit the message that was rejected.

5.3.4.2 Nodal Failures

One of the major differences between the Communication Protocol Model and the Reliability/Availability Model is the simulation of a nodal failure. A network node becomes unavailable by a line outage, mechanical failure or because it has temporarily used all of its buffer space. Two new attributes for each node were added:

STATUS - 0 - down
1 - up

VC.NUM - Number of virtual circuits already allocated

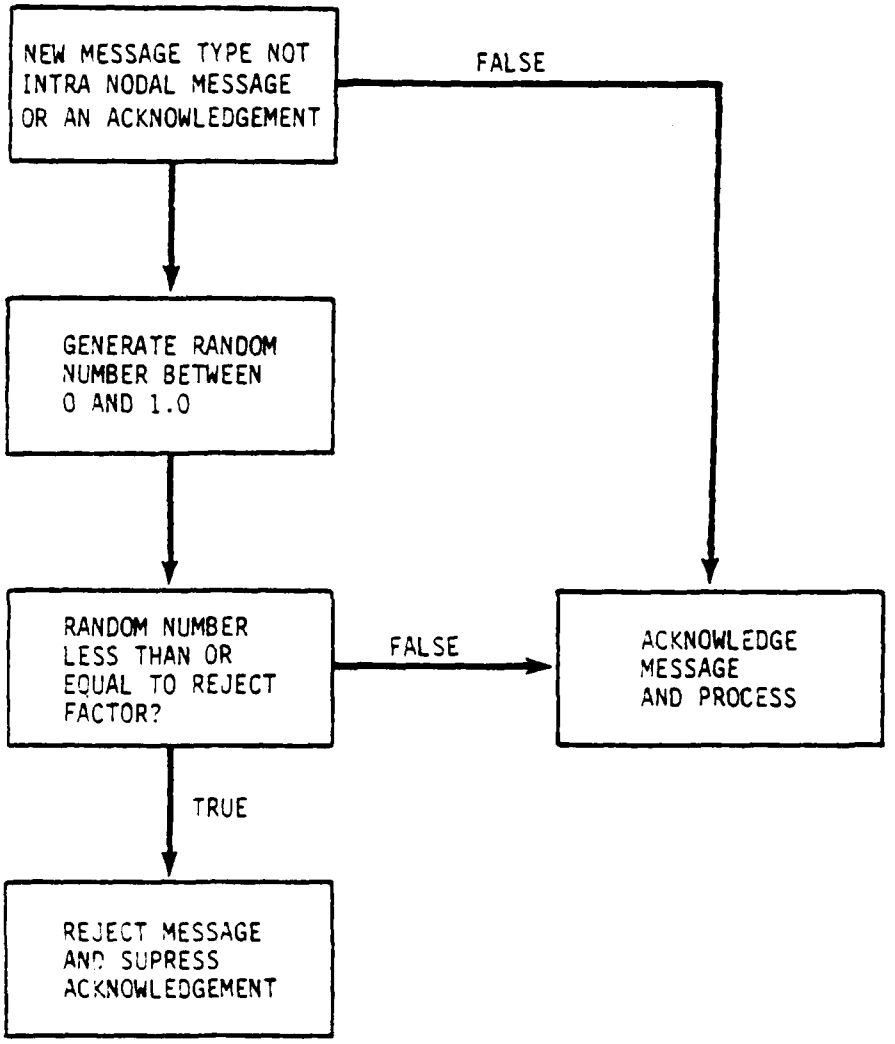



Figure 5.3.4.1-1 Transmission Error Logic

The following user parameters were added to the EXEC file:

MTC - mean time between crashes
DMS - distribution for MTC
MDT - mean down time
DMD - distribution for MDT

Distribution options are:

1 - No change from EXEC file specified value

2 - Uniform distribution

3 - Exponential distribution

For each node an interval (MTC) between crashes will be computed; when that simulation time has elapsed, the status variable will be set to represent a downed site. Another computation using MDT will result in the interval representing the duration of the crash. After the down time has elapsed, the status value will be toggled to represent up. As each virtual circuit is allocated, the number of virtual circuits for any node on the virtual circuit is incremented. As a virtual circuit is released, each node on that virtual circuit has its variable VC.NUM decremented.

5.3.4.3 Adaptive Routing

In the Reliability/Availablity Model a message has a choice of paths that can be travelled to a target node. The user specifies the possible multiple paths that exist via the route file. When a virtual circuit is to be chosen, the VC request routine will build a virtual circuit log one node at a time. Each node added to a virtual circuit will be chosen from alternatives using the following two step criteria:

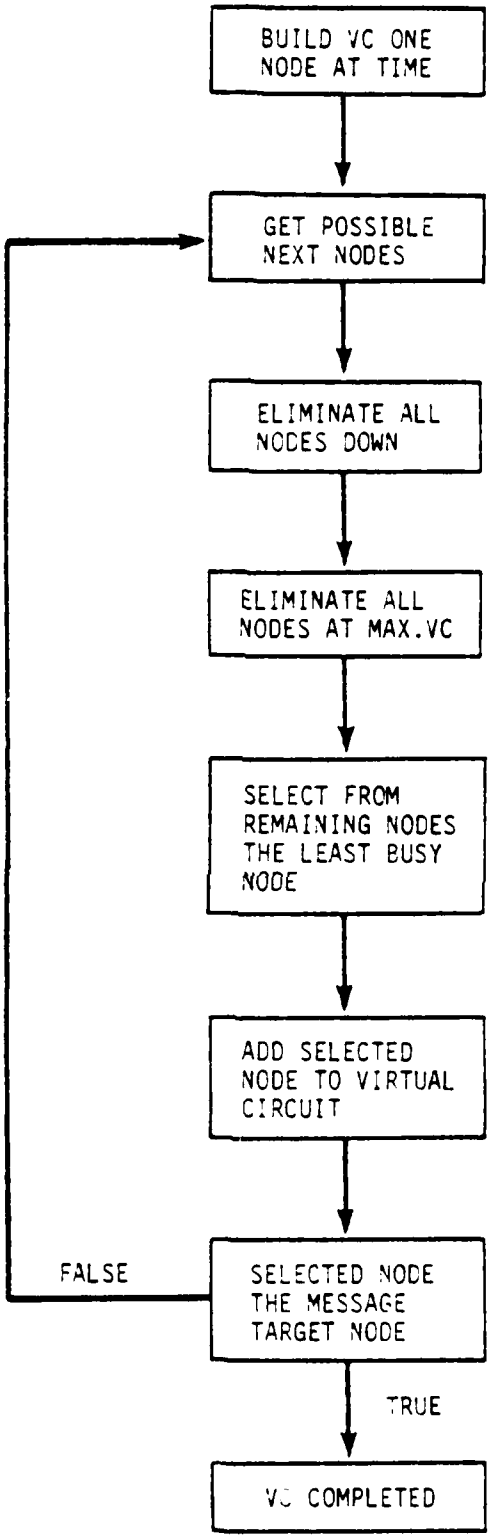



Figure 5.3.4.3-1 Adaptive Routing Functional Flow

1) Eliminate from possible next nodes any nodes down and any nodes already at maximum virtual circuits.

2) Select the node with the least number of virtual circuits allocated.

The second step minimizes network congestion, a similar approach is used in routine VC.NODE when a node in the virtual circuit fails after the message has left the originating node. Thus, the new virtual circuit retains nodes already traversed and appends nodes that are up and least busy. If no valid alternate virtual circuit is possible, the message will be buffered until a virtual circuit becomes available. The function flow chart for this adaptive routing is presented in Figure 5.3.4.3-1

### 5.3.5 SIMULATION OUTPUT

The DSS Simulator produces two output reports:

- Message Statistics
- ECSS Standard Reports
  (See Appendix E for examples )

For a description of each report contained in this section refer to the User's Manual. For a complete sample of these reports refer to Appendix E.

## 5.4 DISTRIBUTED DATABASE MODEL

The DB model provides a tool for the system designer to model and study a proposed Distributed Database Management System (DDMS) during its planning stages. Availability of such a tool is important because performance characteristics of a DDBM System is dependent on many factors. Simulation models can help in the analyses that are necessary to design a cost-effective system that meets certain performance requirements. The DB Model includes these components:

- Transaction Updates/Modifications
- Transaction Retrievals

- Number of Lockups
- Concurrency Control
- File Allocation
- Transaction Rates
- Transmission Overhead
- Processor Utilization
- Buffer Utilization
- Average Queue Lengths at File Servers
- Delays due to Blocking/Synchronization
- Transaction Throughput
- Transaction Response Time

The functional characteristics of the model are such that the user may vary a wide range of DDBM System parameters and study the effects in terms of the resource utilization (channels, processors, buffers, transmission paths) and the net user effect (throughputs, response times).

The DB model will be explained by first describing the Distributed Database Management System Architecture used as a reference to build the three submodels that compose the DB model. Each submodel will then be described in detail. Finally, the DB model implementation using DSS will be covered.

### 5.4.1 DISTRIBUTED DATABASE MANAGEMENT SYSTEM ARCHITECTURE

A Distributed Database Management System (DDBMS) may be viewed as a collection of host sites connected by a communication sub-network. Each site of a DDBMS contains one or both of the following software modules:

- A Transaction Manager (TM) that supervises transactions between users and the DDBMS.
- A Data Manager (DM) that manages the stored Database.

Thus, a DDBMS contains four elements: transactions, TMs, DMs and data (Figure 5.4.1-1). A user's transactions communicate with TMs which communicate with DMs and DMs manage the data. TMs cannot communicate with other TMs nor can DMs communicate with other DMs. [BERN 81].

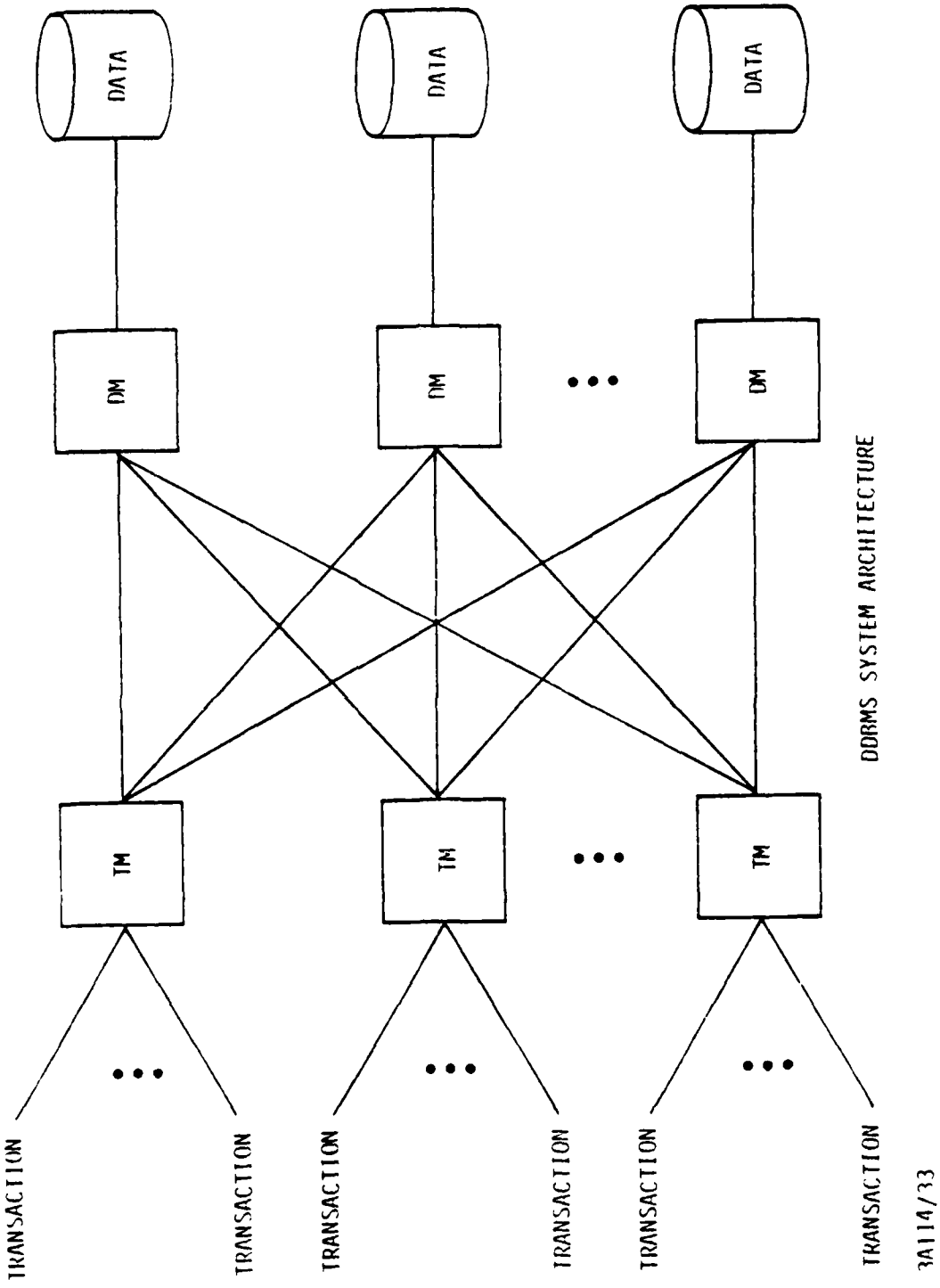



Figure 5.4.1-1 DDBMS System Architecture

This model of a DDBMS assumes perfectly reliable communication links between nodes. In order to make the model more realistic the DB Model used the services provided by the Reliability/Availability (R/A) and the Communication Protocol (CP) Models. In this section, however, we will be concerned primarily with the model elements that are specific to a DDBMS. Models of nodal and transmission failures as well as the X.25 interface model are described in preceeding sections (See 5.1 and 5.2).

The DB model consists of three sub-models.

The three sub models are:

- Database Topology submodel (DT) which simulates distribution of data across a computer network (See 4.4.1.1).
- Transaction Manager submodel (TM) which simulates the supervision of transactions by the TM module of a DDBMS (See 4.4.1.2).
- Data Manager submodel (DM) which simulates the actual management of stored data by a DM (See 4.4.1.3).

As stated before, the Communication Protocol Mechanisms were utilized to transport data and transactions across a packet-switched network. The Reliability/Availability Model's mechanisms of transmission errors and nodal failures were utilized to simulate the real world constraints imposed on any Distributed Database System.

Each of the three submodels will now be described in detail in the following sections.

5.4.1.1 Database Topology (DT) SubModel

The Database submodel contains three major elements

1) File allocation - Which allows the user to map logical files to network nodes.
2) Data granularity - Which provides the user with the flexability to define the level of abstraction. Data can be modeled as an entire database, files, records or even fields within a record.

3) Physical and Logical Database units - Both logical and physical database units can be defined by the modeller. A logical database unit can be declared as residing at only one node (centralized) or replicated at several nodes (decentralized). A physical database unit can be declared as physically located on a particular disk within a nodal site.

5.4.1.2 Transaction Manager (TM) SubModel

A transaction is defined as a sequence of reads and writes (R/W) issued by a database user. The idea of a message used in the previous models (Communication Protocol, Reliability/Availability) has been expanded to incorporate the more general notion of a transaction. Just as parameters were available to declare the dimensions of a message, the following parameters are available to define transactions:

- Mean transaction lengtn - the average number of R/W operators.
- Read/Write Ratio - allows the analyst to declare the proportion of reads to writes. In a distributed database, a read will only access one copy of a file with preference for the local copy (if one exists). A write to any particular file must be exploded to the number necessary to update all file copies existing in a distributed database.
- Originating node - this is the node from which transactions are generated and can be specified by the user.

The software module of a DDBMS that supervises interactions between transactions and the DDBMS is called a Transaction Manager (TM). In the Distributed Database Model an external process named TM is used to model the Transaction Manager. Figure 5.4.1.2-1 is the functional flow chart for the TM external Process. When a message (Type 1) is received by a host from a terminal, external process TM is started. External process TM will perform the following functions:

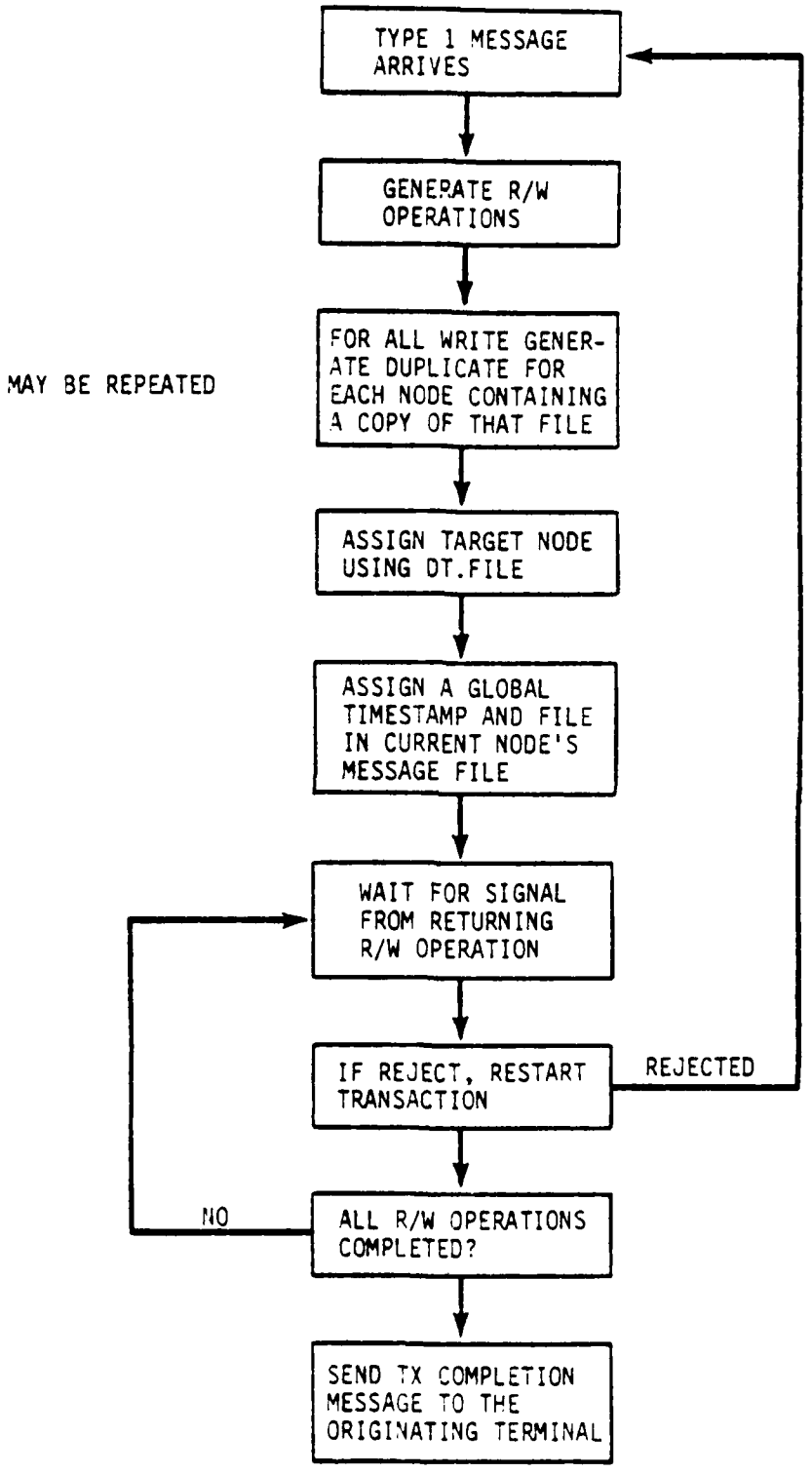



Figure 5.4.1.2-1 Functional Flow Chart Transaction Manager

- Generate the appropriate read and write operations as new messages with target files.
- If a write was generated, duplicates will be created for each node that contains a copy of that target file.
- A database directory lookup will be performed using the declarations from the DT File to obtain the nodal location and disk number for each file. On a read operation, the local file will be used if one exists.
- Each R/W operation is assigned a global timestamp to be used for timestamp ordering.
- Each R/W operation message is filed in the current node's message file.
- The TM then tracks all R/W operations created and upon their return, checks for any rejects. If any R/W operation was rejected, the TM will restart the transaction.
- Upon receipt of all R/W operations originally dispatched, the TM will start JOB HT which signals the originating terminal that the transaction was completed or restarted.

5.4.1.3 Database Manager (DM) SubModel

The Database Manager (DM) is the DDBMS software module that manages the stored database. TMs issue commands to DMs specifying stored database units to be read or written. In a distributed database environment, DMs may receive commands from many TMs (See Figure 5.4.1-1). Because DMs may receive operations from two or more TMs against the same database unit, conflicts may occur. For example, if two different TMs issued simultaneous write operatons against the same database item, a subsequent read operation could result in an inconsistency at different sites depending on the sequence of the write operations.

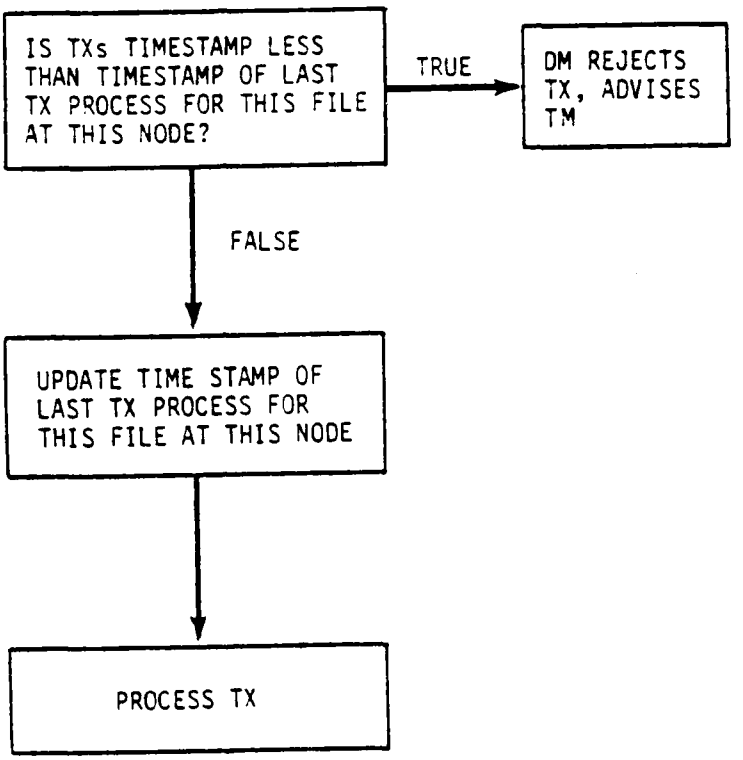
Concurrency control is the activity of preventing interference among transactions that simultaneously access a shared database. It is the concurrency control mechanism that must resolve the conflict between different TM operations against the same data item. The concurrency control algorithm implemented in the DB model is Timestamp Ordering (T/O).

The functional logic flow for the T/O is displayed in Figure 5.4.1.3-1.

The major functions that a T/O scheduler performs are listed below.

1) TMs assign a globally unique timestamp for each transaction.
2) TMs attach the transaction timestamp to each of its R/W operations.
3) DMs must process conflicting operations in timestamp order.
4) If an operation arrives 'late' with a timestamp less than the timestamp of the last successful operation against that data item; the operation is rejected by DM. This rejection occurs before the actual stored data has been impacted to avoid the complexity of rollback.
5) The TM that originated the operation will audit each returning operation. If any operation was rejected by a DM, the TM will restart the transaction [BERN 81].

T/O is a type of schedule control. Schedulers can delay or reject transactons. In a distributed environment schedulers can be centrally located or distributed throughout the network. In the DB model the T/O scheduler was implemented as distributed control residing at each host site and performed by DMs.




2A160/7A

Figure 5.4.1.3-1 Functional LOGIC Flow for timestamp Ordering (T/O) Schedular

### 5.4.2 LAYERED APPROACH TO BUILDING DATABASE MODEL

The CP and R/A models formed the building blocks of the DB model. Since the DB model's domain resided at the application level (layer 7) of the ISO Reference Model (see 5.1), existing R/A and CP modules corresponding to lower ISO levels were utilized without change. The CP model's virtual circuits provided the point-to-point connection for transactions to access remote files while the R/A model's mechanisms for nodal failures, transmission errors and adaptive routing provide a more realistic setting for the Distributed Database Model.

### 5.4.3 MODEL ARCHITECTURE

The DB Model consists of two Model Types:

- A model to describe the behavior of the host sites and
- A model to describe the behavior of switching nodes.

#### 5.4.3.1 Host Sites

The function of the host site is to generate new messages and destroy messages sent from other host nodes. Each host site consists of:

- A host processor
- A Network Interface Processor (NIP)
- A variable number of internodal transmission devices
- 2 Channels
- 1 Buffer
- 10 Terminals
- 10 Disk Drives

The host sites in the Detail Models differ from host sites in HLM Models (Section 4) by allowing hosts to perform both host functions and switching functions. Thus, one model (Model 1 - Host Sites) would have been sufficient to simulate the different complexity levels, however two models - Host and Switch - were retained. The Switching Model is available for simulating nodes performing switching functions exclusively.

Figure 5.4.3.1-1 graphically protrays the Host Site Configuration.

Table 5.4.3.1-1 provides a summary for the System Description for the Host Sites.

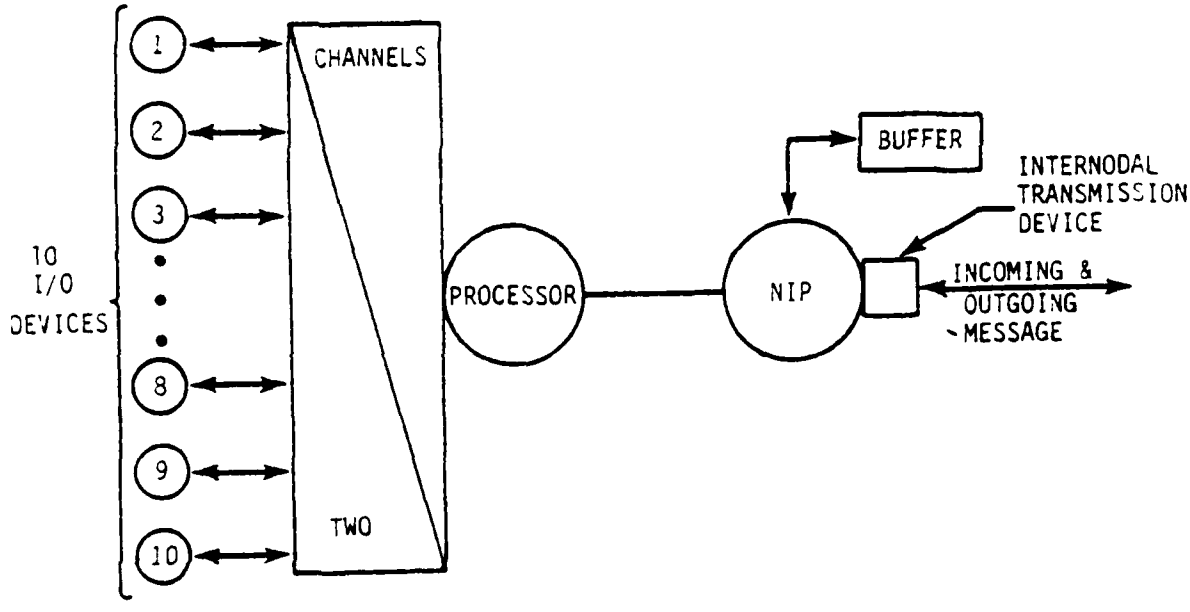



Figure 5.4.3.1-1 Host Site Configuration

Table 5.4.3.1-1 Host Site Configuration

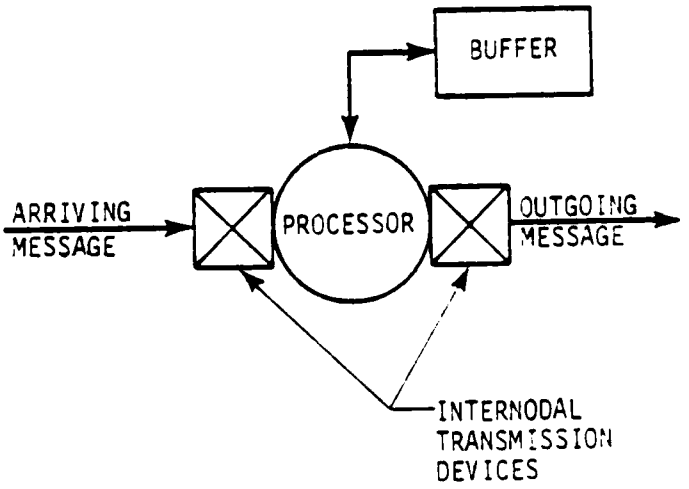
| DEVICE NAME | ECSS DEVICE TYPE | PERFORMANCE SPECIFICATIONS |
|---|---|---|
| D01A.CPU (Processor) | EXECUTIONS ANDS JOB STORE DEVICE | 500,000 INSTRUCTIONS /SEC |
| D01A.NIP (Network Interface Processor) | NETWORK INPUT PROCESSOR EXECUTES AND JOB STORE DEVICES | 500,000 INSTRUCTIONS /SEC |
| T01A.M01 (Internodal Transmission Device) | TRANSMISSION DEVICE | 20,000 BYTES/SEC |
| D01B.CHANNELS (Channel) | TRANSMISSION DEVICE | 20,000 BYTES/SEC |
| D01D.BUF (Buffer) | STORAGE DEVICE | 10,000 BYTES |
| D01C.TERMINAL (Terminal) | I/O DEVICE | 9600 BYTES/SEC |
| P01A.T.PATH (Internodal Path) | PATH | CONNECT D01B.CHANNELS TO D01C TERMINALS |

5.4.3.2 Switching Nodes

The function of the switching node is to accept and forward messages along the network. Each switching node consists of:

- A Processor
- A variable number of Internodal Transmission devices
- A Buffer

The Processor and Buffer declarations are provided in the DSS Switching Model. The specifications for the Internodal Transmission devices are defined in the TP.FILE. Figure 5.4.3.2-1 displays the Switching Node Configuration. Table 5.4.3.2-1 provides a summary for the System Description Section for the switching node.




2A83/6

Figure 5.4.3.2-1 Switching Node Configuration (Dedicated Option)

Table 5.4.3.2-1 System Description for Switching Node

| DEVICE NAME | DEVICE TYPE | ECSS PERFORMANCE SPECIFICATIONS |
|---|---|---|
| D02A.CPU (Processor) | EXECUTION AND JOB STORE DEVICE | 500,000 INSTRUCTIONS/SEC |
| T02A.M01 (Internodal Transmission Device) | TRANSMISSION DEVICES | 30,000 BYTES/SEC |
| D02B.BUF (Buffer) | STORAGE DEVICE | 10,000 BYTES |
| P01A.N01.N02 (Internodal Path) | PATH | CONNECTS T01A.M01 TO T02A.M02 |

### 5.4.4 FUNCTIONAL LOGIC FLOW

Two major external processes provide the control functions (e.g., message generation and starting computer jobs) for the DB model. They are:

- ..ARR process (message generation)
- CP Process (communication process - controlling computer jobs)

A brief description of each process is provided:

Message Generation - External Process ..ARR

The ..ARR process provides the vehicle for generating every message introduced into the network. This process simply cycles according to the interarrival rate (MIT) established in the MESS file or defaulted in the EXEC file. Messages are created each time the process cycles. The number of instances of ..ARR in the External Process Reports indicates the number of different message types generated during the simulation run.

Communication Process - External Process CP

The primary function of the communication process is to control the communication between nodes. A communication process may start jobs on a processing unit which simulates the activity of transferring, switching and receiving messages. Statistics on these external processes are displayed on the External Process Report. The number of instances for the CP processes should be one. These processes are started only once and continue to operate for the full duration of the simulation.

The communication process activates and controls these jobs:

- JOB SM
- JOB RM
- JOB HT
- JOB TH
- JOB HN
- JOB NH
- JOB AK
- JOB PA
- JOB PC
- JOB RC
- JOB RD
- JOB SS

A description of these jobs is provided.

JOB SM - Message Transferring

JOB SM simulates the activity of transfering a message from a current site to the next node. It accomplishes this task by filing the pointer to the message (MSG) data in the switching nodes message file (..MGFILE). The JOB concludes by signaling the switching node. This notifies the communication process that it has a message waiting for it. The total number of instances for a unique JOB SM can be interpreted as the total number of messages transmitted from one site to another.

JOB RM - Retransmit Message

JOB RM simulates the retransmission of a message if an acknowledgement has not been received within a user specified time. The number of instances of JOB RM indicates the number of packets sent from a particular node which were rejected by an adjacent node.

JOB AK- Acknowledgement

JOB AK is responsible for sending acknowledgements for accepted packets. This job is only operational when a "1" is indicated for "ACK" in the EXEC file. The number of instances of JOB AK indicates the number of packets accepted by a particular node.

JOB PA - Packet Assembly

JOB PA simulates the reassembly of a message at the NIP. The number of instance of JOB PA indicates the total number of completely reassembled messages at a host node.

JOB PC - Post Communication

JOB PC is the timing mechanism that schedules the retransmit command for all unacknowledged packets. The user specifies the frequency in which this job is performed by selecting the maximum response time tolerable for all acknowledgements at the host (TMH) and switching nodes (TMC). The number of instances of this job indicates how many times a particular node assessed the status of its outgoing packets.

JOB TH- Terminal to Host Transmission

This job simulates the transfering of a message from a host site's terminal to the Host processor. Tasks accomplished by this job are:

- Sending appropriate bytes from a randomly selected terminal via a randomly selected channel,
- Message type is changed from a new message (Type 1) to 'received at Host', (Type 2),
- Message is then filed in current node's message file,
- CP is signaled.

The number of unique occurances of this job can be interpreted as total messages received at that host from its users.

JOB HN- Host to NIP Transfer

This job increases simulation time to model the overhead incurred by the HOST CPU when relaying a message to its Network Interface Processor (NIP). JOB HN accomplishes this by changing message type to 'received at NIP' (Type 3), filing the message in its message file and signaling CP. Total number of instances for this job for any given node, represents total number of messages entered into network.

JOB NH- NIP to Host Transfer

This job incurs simulation time to represent overhead when messages flow from a Host Site NIP to its Host CPU. Total number of instances for that node represent number of messages received by that node from the network.

JOB HT - Host to Terminal Transfer

This job simulates the transferring of a message from a Host CPU to its terminals. It accomplishes this by sending appropriate bytes via a randomly selected channel to a randomly selected terminal. Total number of instances for this job for any given node represents communication activity between CPU and its terminals.

JOB RC - Reroute for Congestion

This job simulates the selection of a new virtual circuit when the next node in the currect virtual circuit is out of buffer space. Total number of instances represents network rerouting due to traffic overload.

JOB RD - Reroute because of Nodal Failure

This job simulates the selection of a new virtual circuit when the next node in the current virtual circuit is disconnected from the network. Total number of instances represents network rerouting due to nodal failures.

JOB SS - Staggered Signal

This job staggers the signal to the External Process TM (Transaction Manager) to prevent simultaneous (occurring in the same simulation time segment) signals from being interpreted as the same signal.

EXTERNAL PROCESS - RECOV

Controls the time interval a node is unavailable to the network. When the External Process RECOV is signalled after the duration of the crash, the node's status is toggled to 1 (up).

EXTERNAL PROCESS - CRASH

Controls the time interval between failures for a network node. After a time interval has elapsed, the External Process Crash is signaled and the node's status is toggled to Ø (Down).

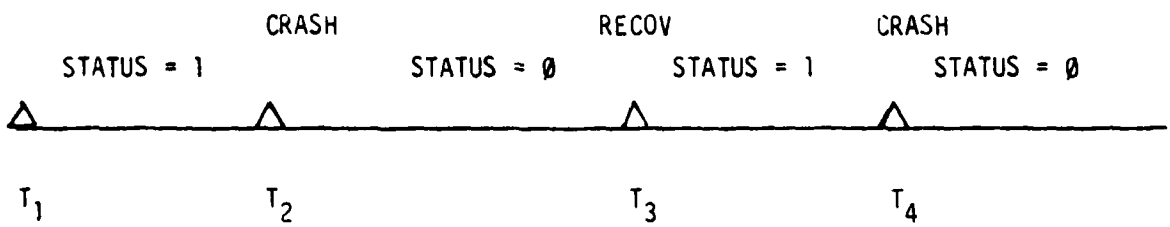



Figure 5.4.4-1 External Process CRASH and RECOV Relationship

Figure 5.4.4-1 shows the relationship between DB model's external process CRASH and external process RECOV (See 5.3.4.2 R/A's Nodal Failures).

One new External Process was added to the DB.

External Process TM - Transaction Manager

This External Process supervises the interaction between transactions and the Distributed Database Management System (DDBMS). Refer to Section 5.4.1.2 for a detailed description of TM.

Section 5.4.1 explained how the Distributed Database Management System Architecture was used as a framework to implement the DB model. Each of the three submodels DT, TM and DM, were described. This section will cover the functional logic flow of the TM to DM interactions. When a transaction arrives at a host, the TM explodes the transaction into individual read and write operations. Each read or write operation is routed to either its host application level or a remote host's application level depending on file location. A read will always be routed to the current node's application level if the target file exists at that site. Writes are duplicated for every node at which the target file is located (See 5.4.1.2). Upon arrival at the appropriate application site's level, the DM checks the operation's timestamp against the timestamp of the last successful operation to access the target file. If the operation is out-of-sequence, the DM rejects the operation and reroutes the operation back to the originating TM. Otherwise, the DM processes the operation, updates the file's last access timestamp and returns the operation to originating TM as successful (See 5.4.1.3).

As the TM receives the returning operation, the TM checks for rejections. If any operation was rejected by a DM, the entire transaction with its original R/W operations is restarted. Once all transactions are received, the TM sends a message to the originating terminal, appropriately advising "transaction completed" or "transaction restarted". Figure 5.4.4-2 displays the Functional Logic flow for the TM-DM interactions.

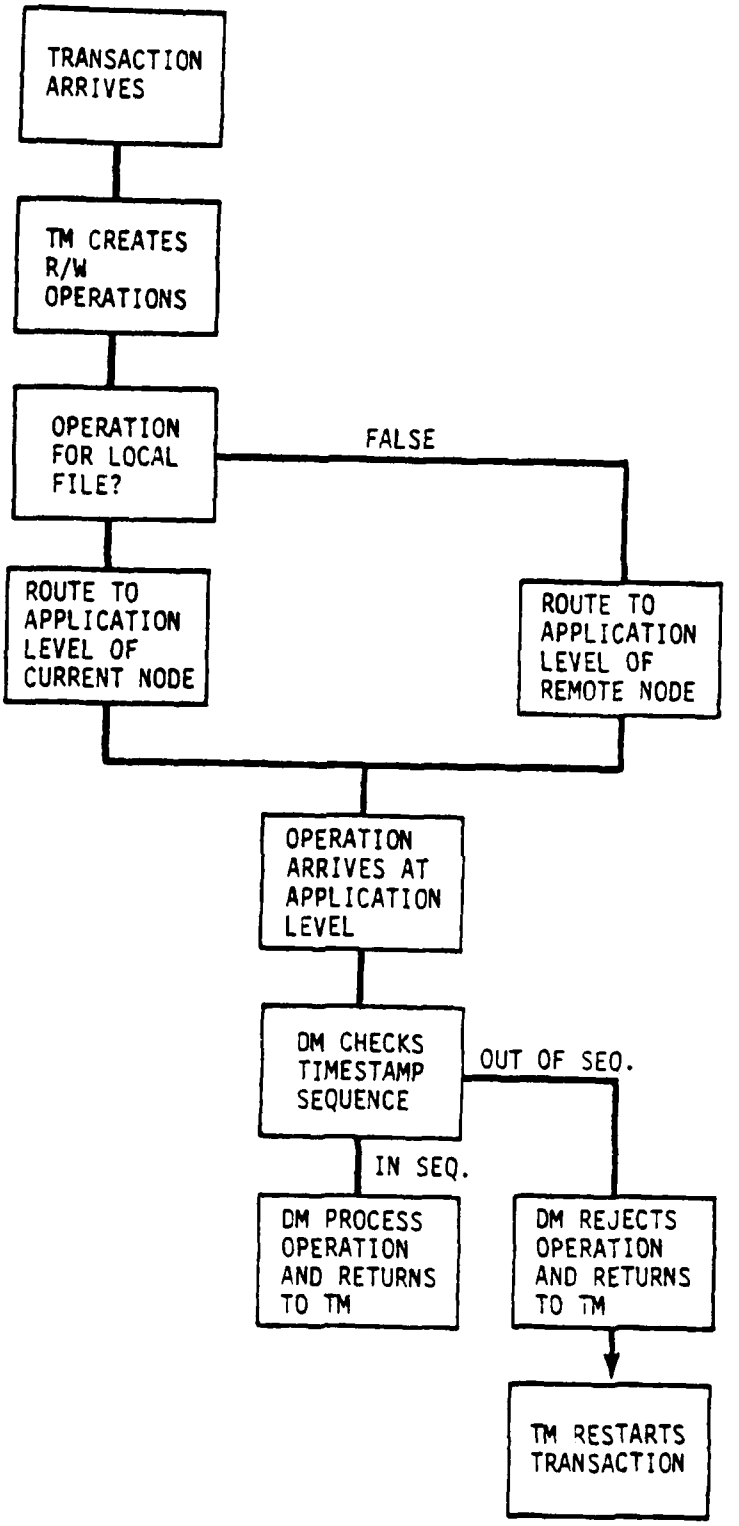



Figure 5.4.4-2 Functional Logic Flow for TM-DM Interactions

### 5.4.5 SIMULATION OUTPUT

The DSS Simulator produces two classes of output reports:

- Transaction Statistics
- ECSS Standard Reports
  (See Appendix F for examples )

For a description of each report contained in this section refer to Section 2.4 of the User's Manual. For a complete sample of these reports refer to Appendix F. A Sample of these reports follows.

MESSAGE STATISTICS

| TYPE | NO. OF MESSAGES | DELAY TIME MIN. | MAX. | AVG. | STD. |
|---|---|---|---|---|---|
| 1 | 10 | .34 | .76 | .43 | .12 |
| 2 | 13 | .05 | .57 | .24 | .17 |

SYSTEM WIDE

| NO. OF MESSAGES | DELAY TIME MIN. | MAX. | AVG. | STD. |
|---|---|---|---|---|
| 23 | .05 | .76 | .33 | .18 |

NODE STATISTICS

| NODE | NO. OF NODAL CRASHES | PERCENT TIME AVAILABLE |
|---|---|---|
| 1 | 7 | 93.000 |
| 2 | 7 | 93.000 |
| 3 | 8 | 92.000 |
| 4 | 6 | 94.000 |
| 5 | 7 | 93.000 |

TRANSACTION STATISTICS

| NODE | NO. COMPLETED | NO. RESTARTED | AVG. RESTARTED |
|---|---|---|---|
| 1 | 0 | 0 | 0. |
| 2 | 0 | 0 | 0. |
| 3 | 0 | 0 | 0. |
| 4 | 0 | 0 | 0. |
| 5 | 0 | 0 | 0. |

S I M U L A T I O N  S T A T I S T I C S
FROM 0. TO 40.000000

--- EXECUTION REPORT ---

| DEVICE NAME | TOTAL EXECUTION TIME | JOB NAME | JOB EXECUTION TIME | % OF DEVICE EXECUTION |
|---|---|---|---|---|
| D01A.NIP | 1.044750 | SM01 | .565000 | 54.080 |
| | | AK01 | .475000 | 45.465 |
| | | MB01 | .004200 | .402 |
| | | RM01 | .000550 | .053 |

| DEVICE NAME | TOTAL EXECUTION TIME | JOB NAME | JOB EXECUTION TIME | % OF DEVICE EXECUTION |
|---|---|---|---|---|
| D03A.NIP | .818500 | SM03 | .405000 | 49.481 |
| | | AK03 | .380000 | 46.420 |
| | | PA03 | .003000 | .367 |
| | | NH03 | .030000 | 3.665 |
| | | RM03 | .000500 | .061 |

| DEVICE NAME | TOTAL EXECUTION TIME | JOB NAME | JOB EXECUTION TIME | % OF DEVICE EXECUTION |
|---|---|---|---|---|
| D04A.NIP | .704250 | SM04 | .330000 | 46.858 |
| | | AK04 | .330000 | 46.858 |
| | | PA04 | .004000 | .568 |
| | | NH04 | .040000 | 5.680 |
| | | RM04 | .000250 | .035 |

| DEVICE NAME | TOTAL EXECUTION TIME | JOB NAME | JOB EXECUTION TIME | % OF DEVICE EXECUTION |
|---|---|---|---|---|
| D05A.NIP | .468000 | SM05 | .305000 | 65.171 |
| | | AK05 | .150000 | 32.051 |
| | | MB05 | .001750 | .374 |
| | | PA05 | .001000 | .214 |
| | | NH05 | .010000 | 2.137 |
| | | RM05 | .000250 | .053 |

00321

SIMULATION STATISTICS
FROM 0. TO 40.000000

--- TRANSMISSION PATH REPORT ---

| PATH NAME | NUMBER OF TRANSMISSIONS TOTAL* | AVERAGE | STD DEV | MAXIMUM | TRANSMISSION LENGTH AVERAGE | MAXIMUM | PERCENTAGE UTILIZATION |
|---|---|---|---|---|---|---|---|
| P05B.D.PATH | 31 | .001 | .028 | 1 | .000984 | .001000 | .076 |
| P04B.D.PATH | 7 | .000 | .012 | 1 | .000786 | .001000 | .014 |
| P03B.D.PATH | 6 | .000 | .010 | 1 | .000667 | .001000 | .010 |
| P01B.D.PATH | 33 | .000 | .020 | 1 | .000500 | .000500 | .041 |
| P05A.T.PATH | 5 | .013 | .113 | 1 | .104167 | .104167 | 1.302 |
| P04A.T.PATH | 0 | 0. | 0. | 0 | 0. | 0. | 0. |
| P03A.T.PATH | 0 | 0. | 0. | 0 | 0. | 0. | 0. |
| P01A.T.PATH | 12 | .016 | .133 | 2 | .052083 | .052083 | 1.445 |
| I05B.N05.N04 | 69 | .057 | .233 | 1 | .033333 | .033333 | 5.750 |
| I05A.N05.N03 | 102 | 085 | .279 | 1 | .033333 | .033333 | 8.500 |
| I02B.N02.N04 | 188 | .157 | .363 | 1 | .033333 | .033333 | 15.667 |
| I02A.N02.N03 | 210 | .175 | .380 | 1 | .033333 | .033333 | 17.500 |
| I01A.N01.N02 | 391 | .326 | .469 | 1 | .033333 | .033333 | 32.583 |

* NOTE* THIS INCLUDES CONTROL OPERATIONS.

--- TRANSMISSION PATH QUEUE REPORT ---

| PATH NAME | TOTAL REQUESTS | AVG WAIT FOR TRANSMISSION | TOTAL ENQUEUED | QUEUE LENGTH AVG | STD DEV | MAX | TIME IN QUEUE AVERAGE | MAXIMUM | % TIME EMPTY |
|---|---|---|---|---|---|---|---|---|---|
| P05B.D.PATH | 31 | .000000 | 5 | .000 | .000 | 1 | .000000 | .000000 | 100.000 |
| P04B.D.PATH | 7 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| P03B.D.PATH | 6 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| P01B.D.PATH | 33 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| P05A.T.PATH | 5 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| P04A.T.PATH | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| P03A.T.PATH | 0 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| P01A.T.PATH | 12 | 0. | 0 | 0. | 0. | 0 | 0. | 0. | 100.000 |
| I05B.N05.N04 | 69 | .019531 | 31 | .034 | .234 | 3 | .043473 | .084500 | 97.657 |
| I05A.N05.N03 | 102 | .057746 | 54 | .147 | .847 | 7 | .109075 | .231517 | 95.784 |
| I02B.N02.N04 | 188 | .027969 | 112 | .131 | .502 | 4 | .046947 | .123333 | 91.951 |
| I02A.N02.N03 | 210 | .045070 | 152 | .237 | .785 | 6 | .062267 | .184717 | 89.126 |
| I01A.N01.N02 | 391 | .207966 | 348 | 2.033 | 4.166 | 24 | .233663 | .784900 | 71.743 |

--- TRANSMISSION REPORT ---

| DEVICE NAME | NUMBER OF TRANSMISSIONS TOTAL | AVERAGE | STD DEV | MAXIMUM | TRANS CAPACITY | PERCENTAGE UTILIZATION | -- TRANSMISSION LENGTH -- AVERAGE | MAXIMUM | % TIME IDLE |
|---|---|---|---|---|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| D01A.NIP | 505 | .002069 | .005000 | 2.612 |
| D03A.NIP | 334 | .002451 | .005000 | 2.046 |
| D04A.NIP | 284 | .002480 | .005000 | 1.761 |
| D05A.NIP | 226 | .002071 | .005000 | 1.170 |
| D01B.CPU | 222 | .002411 | .005000 | 1.338 |
| D03B.CPU | 12 | .000500 | .001000 | .015 |
| D04B.CPU | 14 | .000500 | .001000 | .017 |
| D05B.CPU | 143 | .001628 | .005000 | .582 |
| D02A.NIP | 787 | .002505 | .005000 | 4.928 |

--- EXECUTION (READY) QUEUE REPORT ---

| DEVICE NAME | NUMBER OF ARRIVALS | WAIT FOR SERVICE AVERAGE | WAIT FOR SERVICE MAXIMUM | QUEUE LENGTH AVG | QUEUE LENGTH STD DEV | QUEUE LENGTH MAX | % TIME EMPTY |
|---|---|---|---|---|---|---|---|
| D01A.NIP | 505 | .001358 | .035000 | .017 | .188 | 7 | 98.773 |
| D03A.NIP | 334 | .001327 | .010500 | .011 | .114 | 3 | 98.992 |
| D04A.NIP | 284 | .001594 | .023049 | .011 | .140 | 6 | 99.108 |
| D05A.NIP | 226 | .002455 | .021250 | .014 | .216 | 7 | 99.337 |
| D01B.CPU | 236 | .018386 | .053100 | .108 | 1.020 | 15 | 98.665 |
| D03B.CPU | 12 | 0 | 0. | 0. | 0. | 1 | 100.000 |
| D04B.CPU | 14 | 0. | 0. | 0. | 0. | 1 | 100.000 |
| D05B.CPU | 143 | .021145 | .046250 | .076 | 1.042 | 19 | 99.422 |
| D02A.NIP | 787 | .001822 | .035000 | .036 | .281 | 7 | 97.364 |

--- JOB STORE REPORT ---

## REFERENCES

[BERN 81] Bernstein, P.A., Goldman, N., Concurrency Control in Distributed Database Systems, ACM Computing Surveys, Vol. 13, No. 2, June 1982, pp. 185-221.

[FEDE 76] Federal Computer Performance Evaluation and Simulation Center, Washington D.C., User's Guide - ECSS II, Extended Statistics and Trace Features, Ver. 1.1, August 1976.

[FOLT 80] Folts, H.C., X.25 Transaction - Oriented Features - Datagram and Fast Select, IEEE Transactions on Communications, Vol. Com-28, No. 4, April 1980, pp. 496-499.

[GREE 80] Green, P., An Introduction to Network Architectures and Protocols, IEEE Transactions on Communications, Vol. Com-28, No. 4, April 1980, pp. 413-420.

[HUTC 73] Hutchinson, G.K., The use of Micro Level Simulation in the Design of a Computer Supervisory System, Proc. ACM Symp. on the Simulation of Computer Systems, June 1973, pp. 243-254.

[KIVI 73] Kiviat, P.J., Markowitz, H.M., Villanueva, R., Simstcript II.5 Programming Language, CACI, Los Angeles CA, 1973.

[KOSY 75] Kosy, D.W., The ECSS II Language for Simulating Computer Systems, Rand, Santa Monica CA, 1975.

[MACD 67] MacDougall, M.H., Simulation of an ECS-Based Operating System, SJCC, 1967, pp. 735-741.

[MACD 73] MacDougall, M.H., MacAlpine, J.S., Computer Simulation with ASPOL, Proc. Sym. on Simulation of Computer Systems, June 19-20, 1973, p. 103.

[PRIC 77] Price, W.L., Data Network Simulation: Experiments at the National Physical Laboratory, 1968-1976, Computer Networks, Vol. 1, No. 4, May 1977.

[RUSS 82] Russell, E.G., Building Simulation Models with Simscript II.5 *(Preliminary Printing)*, *CACI Inc., Los Angeles, 1982.*

[RYBC 80] Rybczynski, A., X.25 Interface and End-to-End Virtual Circuit Service Characteristics, IEEE Transactions on Communications, Vol. Com-28, No. 4, April 1980, pp. 500-510.

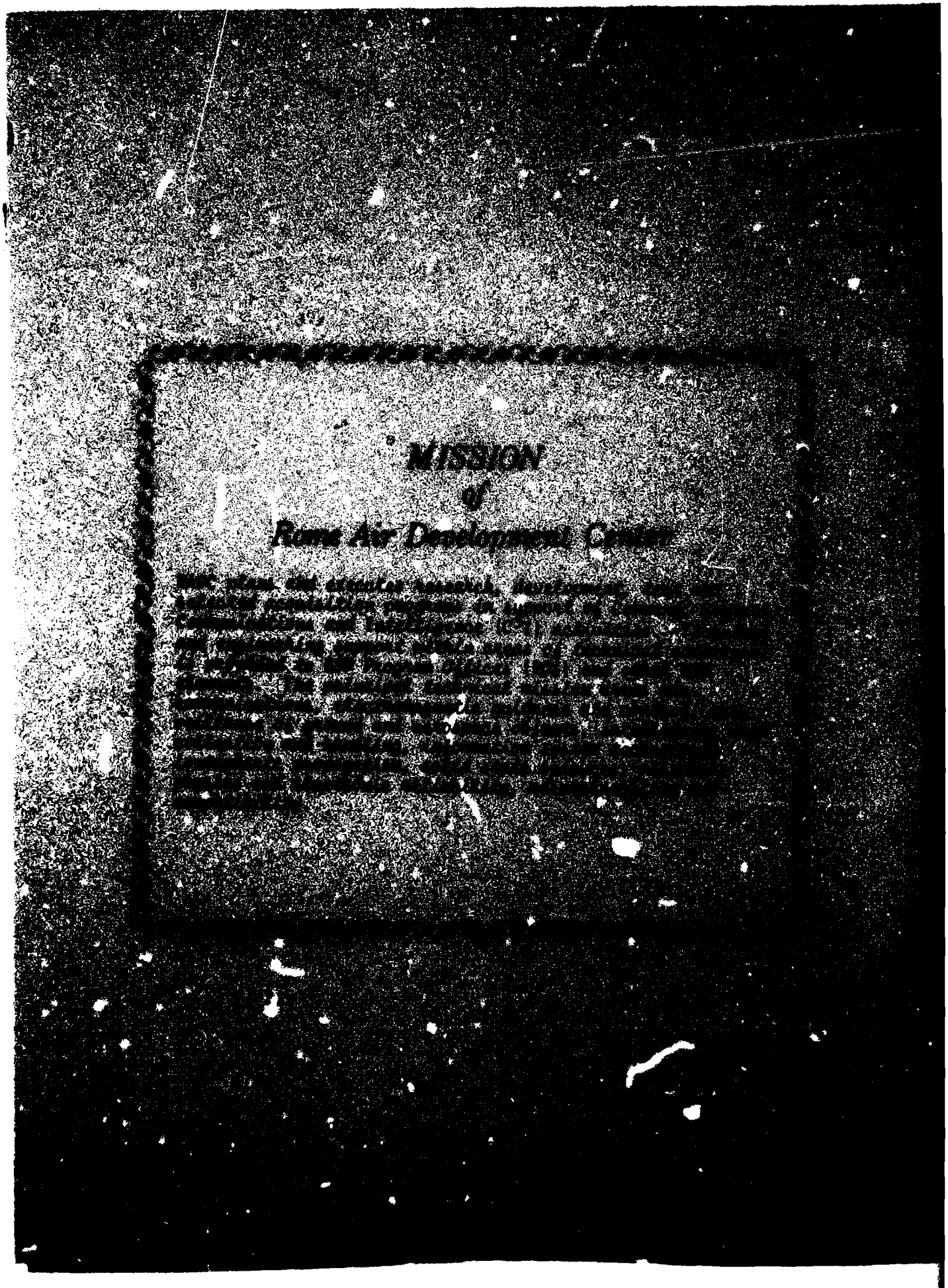
[SCHN 78] Schneider, G.M., A Modeling Package for Simulation of Computer Networks, Simulation, Dec. 1978, pp. 181-192.

[SEAM 69] Seaman, P.H., Soucy, R.C., Simulating Operating Systems, IBM Systems Journal, Vol. 8, No. 4, 1969, pp. 264-279.

[SHOE 78] Shoemaker, S., Editor, Computer Networks and Simulation, North Holland, New York, Sept. 1978.

[TOBA 80] Tobagi, F.A., Multiaccess Protocols in Packet Communication Systems, IEEE Trans. on Comm., Vol. Com-28, No. 4, April 1980, pp. 468-488.

[UNGE 78] Unger, W., Programming languages for Computer System Simulation, *Simulation, April 1978, pp. 101-110*



MISSION
of
Rome Air Development Center